CHI Systems, Inc.
100 Burns Place
Goleta, CA 93117
(805) 984-8868

OK - DTIC

85 - 0219

CHI-5 ARRAY PROCESSOR SUPPORT

Final Technical Report

April 1985

PRINCIPAL INVESTIGATOR: Dr. Glen J. Culler

PROJECT SCIENTISTS: Dr. Judith B. Bruckner
Thomas W. Fuller
Virginia R. Grant
Dr. Michael McCammon
Dr. Jean A. Nisbet

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3625. Contract MDA 903-82-C-0136.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

INTRODUCTION

The CHI-5, a low-cost, generalized array processor, was developed under an earlier DARPA contract, MDA903-78-C-0313 as part of an effort to develop low-cost packet speech hardware. The CHI-5 was used successfully by CHI Systems, Inc. and SRI International for digital voice coding using linear predictive coding (LPC) for transmission through packet- switched networks.

In support of the DARPA program in packetized speech, the CHI-5 could be used not only for the basic LPC algorithms, but also to support development and demonstration of algorithms for degraded speech environments, and integration of voice and data transmissions. In addition, the CHI-5 could serve as a low-cost compute server, attached to a host computer, to support circuit simulation in programs developing other low-cost speech hardware.

For the CHI-5 to be useful to a variety of DARPA contractors involved in these programs, it was necessary to provide generally usable program development software for the CHI-5 and to make additional processors available. The CHI-5 host interface needed to be revised to make it possible to connect it to host computers with a standard device interface. Also, a standard asynchronous interface was needed to allow direct connection to communication media and for other simple, low bandwidth interface applications. This contract, MDA-903-82-C-0136, was established to meet these needs by providing ten additional CHI-5 processors with the necessary interfaces and providing spare parts and support for them. It also provides for the development of program development software written in a portable language so that it could be used on computers available at the DARPA contractors who would be using the CHI-5s.

This report describes the hardware and software which was developed during this contract. Section 1 is a description of the hardware enhancements of the CHI-5 and the construction of the additional ten units. Section 2 describes the program development software, including a micro-code assembler and linker, a micro-instruction simulator, and a macro-language assembler and linker. Section 3 describes the interface program to support control and data transfer protocols for use of the serial interface for digital voice, data and control transfers. The appendix contains the micro-code developed for the system and a listing of the serial interface control program.

1. HARDWARE ENHANCEMENTS

In order to make the CHI-5 processor generally useful in a variety of environments, three external interfaces were required. The first of these is a parallel interface to a controlling host computer. This interface allows for rapid transfer of data through DMA access to the data memories of both the host and the CHI-5. It also provides for direct control of the CHI-5 from the host computer by providing for initialize and interrupt controls. The parallel host interface was part of the original design of the CHI-5 and its predecessor, the LPCAP. However, the host side of this interface was originally developed using custom cards which plugged into either a UNIBUS[1] or QBUS[1] chassis of a DEC computer. The interface had to be redesigned to use standard interface cards available from DEC and others.

The second interface, also a part of the CHI-5 as originally designed, provides for simultaneous input and output of analog data. This analog interface supports speech data input and output at 8Khz sampling. No changes were required in this interface.

The final external interface supports a pair of asynchronous serial lines, each operating at programmable rates up to 19.2 K baud. These serial interfaces, although they support relatively slow transfer rates, allow connection of the CHI-5 to almost any computer or terminal as well as a large variety of other equipment, and allow communication over inexpensive, long-distance lines. This serial interface was not provided in the original design, but could be accomodated within the existing I/O and interrupt architecture of the CHI-5.

1.1. Parallel Host Interface

The parallel host interface was redesigned to use signals available from the DR11-B or DRV11-B, interface cards for the UNIBUS and QBUS, respectively. These cards support the host half of a direct memory transfer, either from or to the CHI-5, maintaining the word count and host memory address counter. The CHI-5 host interface card (HIF) uses a three bit function code provided by these interfaces to select one of eight address registers for the CHI-5 data memory address. Upon completion of the transfer, an interrupt signal is generated by the DEC interface card for the host computer, and by the HIF for the CHI-5.

Firmware in the CHI-5 uses a function code of 7 to indicate that the data transferred to the CHI-5 is a command, and processes the command immediately. Since each function code can have a separate address in CHI-5 memory for its data, several processes can be set up at once in the CHI-5, each waiting for a signal that their data buffer has been transferred to initiate the next stage in their processing. Digital voice coding, for example, uses two buffers, one holding LPC parameters which have been computed from analog input for transmission or further processing by the host, the other holding parameters received from the host which are to be used by the CHI-5 to synthesize speech for analog output.

The interrupt signals from completion of data transfers are used to initiate parts of the LPC analysis and synthesis programs.

The data transfers to or from the host computer must always be started by the host, who has control over the interface card. However, the program running in the CHI-5 must be able to signal the host when it has data for it, or needs more data to continue, if it is to be able to do real-time processing with speech data. The DMA transfer interface cards available from DEC do not provide for an asynchronous interrupt signal from the attached device, so a second card, the DR11-C or DRV11, is used to provide for this interrupt, as well as a status code which reflects which buffer is ready for a transfer. In addition, the DR11-C provides a means for the host to send an initiate signal to reset the CHI-5 when desired.

1.2. Asynchronous Serial Interface

The asynchronous serial interface of the CHI-5 includes two separate RS232-C standard ports, one configured to connect to a modem or as a terminal to another computer (DTE) and the other configured to connect a terminal to it directly (DCE). The transfer rates of each port can be set separately to one of sixteen speeds. The serial interface uses USART devices which assemble and disassemble characters as they are received or transmitted serially. For each character, an interrupt is generated to the CHI-5, which must then take any input character or provide the next character for transmission. The circuits required for the asynchronous interface are packaged on the arithmetic control unit card of the CHI-5 processor and are connected by a ribbon cable to two D-25 female connectors mounted on the back of the chassis.

[1] UNIBUS and QBUS are trademarks of Digital Equipment Corporation (DEC)

2. PROGRAM DEVELOPMENT SOFTWARE

The CHI-5 architecture supports two levels of instruction decoding The lowest level is a wide micro instruction with separate fields within the instruction to control each part of the hardware for one fixed instruction clock period. Micro instructions are fetched from a program memory, separate from the data memory. Each instruction word is 80 bits long. The program memory includes 2048 words of programmable read only memory, for fixed firmware, and 1024 words of writeable memory. Programming at the micro instruction level allows maximum utilization of the computer, but it requires a fairly detailed understanding of the architecture.

The second type of instruction is called a macro instruction. A macro instruction contains an operation name and a list of operands. It is fetched from the same memory used for data, and is one or more 32 bit doublewords long. Each macro instruction causes the execution of a sequence of micro instructions. A standard macro instruction set is supported using firmware in micro programs in ROM. Application specific macro instructions can be defined by writing their micro programs.

To support this two level programming environment, two separate assemblers and a micro instruction simulator are provided:

CHI5ASM
: A FORTRAN-77 program which generates relocatable, microcode modules for the CHI-5. Separately assembled modules may be linked together by the program CHI5LINK to form a single program module. The *CHI-5 MICRO-PROGRAMMING REFERENCE MANUAL* provides a detailed description of the CHI-5 microinstruction set, micro assembler and linker.

CHI5SIM
: A FORTRAN-77 program which simulates the operation of the CHI-5 hardware at the micro instruction level. This simulator can use the program module generated by CHI5LINK. The *CHI-5 SIMULATOR REFERENCE MANUAL* describes the simulator program in detail.

MACASM
: A FORTRAN-77 program which generates relocatable, macrocode modules. Separately assembled modules from MACASM may be linked together with program modules containing microcode from CHI5LINK by the program MACLNK to create a executable load module. The *CHI-5 MACRO PROGRAMMING REFERENCE MANUAL* provides a detailed description of the macro assembler and linker.

2.1. CHI-5 Micro Assembler and Linker

The microcode assembler allows micro routines to be constructed using symbolic labels, expressions to generate numeric values and mnemonic keyword based definitions for the micro instructions. Pseudo-operations are supported to allow definition of entry points to the routine, references to externally defined symbols,

and to define labels representing the value of an expression. Other pseudo-operations control listing and output options for the assembler.

The language chosen for specification of the micro instructions is as high level as is compatible with full expression of the capabilities of the machine. Each CHI-5 micro instruction is composed of one or more individual operations. An operation is described in terms of an operator with zero, one or two operands and possibly a destination. The operands, operators and destinations correspond to elements of the CHI-5 hardware. Each operation can involve several control fields of the CHI-5 micro instruction, and there is often more than one way to perform a given operation. The assembler automatically selects the hardware elements needed to perform the operation specified and generates the proper control fields, although the programmer can explicitly state what hardware elements are to be used when necessary. It also keeps track of the usage of the control fields and attempts to select alternate data paths or arithmetic elements in order to successfully accommodate the operations specified for the micro instruction. If it cannot succeed, it reports the fields where a conflict in usage has occurred.

The syntax of the individual operations has been chosen to be as natural and simple as possible. Two operand operations, such as most adder and all multiplier operations, are specified using an infix notation, e.g.

*X * V*

MPL + U → U

MPLMPR + TU → UV.

Specific selection of hardware elements or options to operations are given by following the operation by a colon (:) and the option, e.g.

*X * Y:PP* (Unsigned multiplication)

MPLMPR + TU :FG → UV (Use F and G adders instead of G and H adders).

Data transfer operations are specified by giving the source, a right arrow, and the destination. Whenever possible, busses will be used in preference to adders, but if a bus is already in use, or the only path is through an adder, the adder will be used automatically. However, if a path through a specific adder is required, as when a test is to be performed on the value being moved, the adder can be specified.

V → W (uses YBS if available, otherwise the H adder)

T → U (uses either the F or G adder)

T:G → U (uses the G adder)

Single operand operations, such as INC or DEC, use an operator, operand syntax:

INC XA; CLR S; GOTO label1.

Operations requiring no operands are specified by their mnemonics alone:

INT HOST; RTN; READ.

The assembler output is a relocateable object module containing the information needed to combine it with other similar modules into a load module. CHI5LINK

is an interactive program, also written in FORTRAN-77, which combines these object modules, resolving references between separate modules, to build a single load module with instructions located at fixed absolute addresses in the program memory. CHI5LINK also produces a symbol table, giving the entry point address associated with each microprogram. The symbol table and load module are used by the simulator for testing of the microprograms. The symbol table is used by the macro assembler to allow it to assign operation values to macro instructions which will use these microprograms. If the load module is linked to load into writeable program memory, the macro linker can include the load module in its data memory image to make it available for loading into program memory under control of the macro program.

2.2. CHI-5 Microinstruction Simulator

The CHI-5 simulator is an interactive program, written in standard FORTRAN-77, which simulates the operation of a CHI-5 processor. The simulator maintains a functional model of the CHI-5. This model is composed of variables, known as "state variables", which hold values corresponding to values held by the elements in a real CHI-5. A set of commands is available to the user with which the values of state variables may be manipulated. Values may be changed directly, or through the simulated execution of CHI-5 instructions.

The simulator may be run either interactively from a terminal, or as a batch job using a file of commands. Its image of the state of the simulated CHI-5 can be examined, and saved or restored to files. A data file represents the state of the data memories, including the array memories X and Y, the table memory R and the main data memory D. A program file contains the state of the program memory; it is often the output of the micro program linker. A state file holds the simulator state variables, including the contents of the simulated CHI-5 registers. During the simulation of CHI-5 instructions, selected state variables can be traced. These variables are written to a logging file each time an instruction is executed.

The simulator supports the analog input and output devices as real-time I/O by maintaining a clock for analog I/O that 'ticks' once every 500 processor instructions during simulation. This corresponds to the 8 KHz analog sampling rate used by the real device. Analog input and output use files which contain the values for input or hold the result. Host DMA transfer is simulated by LOAD and STORE commands for D-memory without affecting the DA registers, S bits or interrupts. The S bits are set by the user to simulate interrupts when desired.

2.3. CHI-5 Macro Assembler and Linker

At the most basic level, the CHI-5 executes micro instructions, with each micro instruction specifying parallel operations for individual hardware elements. Control of the system, however, including applications, interrupt routines, and commands from a host processor, are specified in a higher level language, the instructions of which are called macro instructions.

A macro instruction consists of an operation code and a list of operands. The macro instructions are fetched from D memory for execution. Each macro involves the execution of two micro programs. The first, called the mode, generally fetches the first operand into a register and reads the next doubleword, if any, of the instruction from D memory. It is usually consists of only one or two instructions and is always located in read only program memory; the upper 5 bits of the operation code specify which mode program is to be used. The second micro program performs the actual operation and may be any length; the lower 11 bits of the operation code give the starting address of the program.

The macro assembler generates macro programs for execution on the CHI-5. In order to do this it must know about the set of instructions which will be available in microprogram memory. This information is provided by macro instruction definition statements which associate a macro instruction with an entry point in a microprogram load module and describes its operands. The assembler uses this information to check the use of each instruction and can generate the proper mode for many instructions depending on the location of the first operand.

In order to allow the assembler to generate the proper mode for variable mode instructions, and check that the operands used are correct for the particular instruction, the assembler associates a type with each label used. It also allocates space in X, Y and D memories if desired and supports data statements for initializing D memory variables.

The output of the CHI-5 macro assembler is a relocateable object file. These files are combined using the macro linker, MACLNK, to make a load module file which is ready for transfer into D memory and execution. The linker is an interactive program similar to CHI5LINK, the micro program linker. It also supports the inclusion of microcode from a CHI5LINK output file in the load module and automatically generates a macro subroutine which will load the file into program memory when it is called.

3. SERIAL LINE CONTROL PROGRAM

As part of the packet speech effort, CHI has implemented a preliminary version of the "NSC Low-Rate Vocoder Interface" to provide a means of connecting the CHI-5 to hosts via its RS-232 serial interface. The CHI version of this protocol provides for transfer of both speech and data between the CHI-5 and an external processor, as well as limited control over the vocoder. This protocol has been used with the CHI-5 at the majority of the contractor sites where the processors have been delivered.

The serial protocol encodes both speech and data into characters for transfer over the serial interface using ASCII character codes between "space" and "_" to represent six bits of information by adding the code for "space" as a number to the six bits of data to get the code to send. This uses half of the available 128 codes. The first 32 codes are not used for information transfer, since they are conventionally used for control information. 31 of the last 32 codes,

corresponding to ASCII characters "`", "a-z", "{","|","}",and "~ " are used to define control information within the protocol. The CHI implementation uses six control characters to provide the following functions:

Address (a)
: Set the data transfer buffer address to the value given by the following three characters.

Data (d)
: Store the data which follows in D memory, at the data transfer buffer address. Eight characters provide enough data for three 16-bit words. The data transfer continues into consecutive addresses in the buffer until another protocol control character is received.

Output data (o)
: Convert three 16-bit words at the current buffer address to eight characters and transmit them. The buffer address is then incremented by three so subsequent output data requests will cause the following data to be sent.

Freqn (f)
: Start analysis of speech received on the A/D interface. Each 20.5 msec a parcel of speech parameters, consisting of a playn (p) character and 48 speech data bits packed into 8 characters, will be transmitted from the CHI-5.

Playn (p)
: Synthesize and output through the D/A interface speech using the following eight characters as data for one 20.5 msec frame.

Stop (s)
: Stop analysis of speech. Quit sending speech parameters.

In addition to the protocol control characters, the CHI implementation used the ASCII control characters XON and XOFF to limit the rate at which parameters are received to the real time analog output rate. It also recognizes XON and XOFF on input and suspends or resumes transmission of speech and data parameters as requested.

This protocol is implemented in the CHI-5 by an interrupt routine which responds to interrupts from the serial lines and two subroutines accessed by the vocoder programs for parameter sending and receiving. Appendix B contains a listing of the macro language program for this protocol.

REFERENCES

1. Bruckner, J. B.,
   CHI-5 MICRO-PROGRAMMING REFERENCE MANUAL, Quarterly Technical Report, MDA 903-82-C-0136-Q1, CHI Systems Inc., Goleta, California, April 1982.

2. Fuller, T. W.,
   CHI-5 SIMULATOR REFERENCE MANUAL, Quarterly Technical Report, MDA 903-82-C-0136-Q2, CHI Systems Inc., Goleta, California, August 1982.

3. Grant, V. R.,
   CHI-5 MACRO-PROGRAMMING REFERENCE MANUAL, Quarterly Technical Report, MDA 903-82-C-0136-Q3, CHI Systems Inc., Goleta, California, October 1983.

4. Culler, G. J. ,
   ACOUSTICAL ARRAY PROCESSOR DESIGN, Final Technical Report, MDA 903-78-C-0313, CHI Systems Inc., Goleta, California, June 1983.

APPENDIX A

Listings of the following CHI-5 microprogram modules are included here:

andl.mic
anm255.mic
asline.mic
ave.mic
bldpop.mic
blkshifts.mic
coding.mic
daops.mic
decim.mic
divbyy.mic
doint.mic
dop.mic
dotxy.mic
exec.mic
extrem.mic
fadd.mic
fcos.mic
fdivs.mic
fexp.mic
fft1.mic
fft2.mic
fftpass.mic
fftrl.mic
filter.mic
finv.mic
float.mic
flog.mic
flts.mic
fltsops.mic
fmult.mic
fsqrt.mic
fsubt.mic
initchk.mic
intsrv.mic
latred.mic
ldfx.mic
ldstf.mic
logpwr.mic
movecx.mic
moverd.mic
moves.mic
multl.mic
ormod.mic
power.mic
radian.mic
randoms.mic
record.mic
rmoves.mic
save.mic
sched.mic
score.mic
sflt.mic
shortops.mic
smvxyd.mic
stepn.mic
stkops.mic
tsttol.mic
upchan.mic
xymadd.mic
xymoves.mic

```
        TITLE ANDL
        ENTRY ANDL
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO ANDL, XYA(DESC)  DESC=XYA,N
" X(XYA+i) = X(XYA+i) .AND. Y(XYA+i)  for i=0,...,N-1

ANDL:   Y->J, X->YA, X->YC
        XC->XA, DEC J, Y->V                     "V=first mask
ANDLP:  X AND V->V, INC YA                      "X .AND. Y->V
        V->X, INC XA, Y->V, DEC J,              "store result
              IF J>0 GOTO ANDLP                 "Repeat
        READ, GOTO EXMAC                        "exit
        END
```



```
        TITLE ANM255
        ENTRY M255TOL, LTOM255
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO M255TOL, DA  DOP=DY->W  Read 32 words and convert
"MACRO LTOM255, DA  DOP=DY->W  D/A output

        EQU DI='12'O, ANALOG='10'O, S='8225'D, SINV2=4

M255TOL: W->DA(DI)                              "set up output address
        ANALOG->DEV                             "select ANALG IO
        '30'D->J                                "Count-2
        -1->W                                   "1's compliment mask
        IO XOR W->V, S->YL                      "first input
        V->U, '160'O AND V->V                   "V=16*n
        '10000'O->XL, '10000'O*V:PP             "shift to get n
        '17'O AND U->V, 0->T, U:G               "V=m, test sign
LOOP:   ML+T+GCO:F->U, '16'D-V:G,               "u=n+1
              '16'D+V:H->V                      "v=m+16
        SHIFT(U)*V:PP,                          "(m+16)*2**(n+1)
              IF G<0 GOTO NEG                   "sign neg?
        '33'D->U                                "2*16.5
        MR-U->U, IO XOR W->V                    "U=val,V=next pt
L2:     U*YL:FR, V->U,                          "VAL*SCL,save code
              '160'O AND V->V                   "clean off 16*n
        XL*V:PP, 0->T                           "shift to get n
        ML+T:F->T, '17'O AND U->V               "T=result,V=m
        WRITE T, U:G, 0->T, DEC J,              "store result
              IF J>0 GOTO LOOP                  "test sign,done?
        ML+T+GCO:F->U, '16'D-V:G,               "U=n+1
              '16'D+V:H->V                      "V=m+16
        SHIFT(U)*V:PP,                          "(m+16)*2**(n+1)
              IF G<0 GOTO NEGL                  "sign negative?
        '33'D->U                                "2*16.5
        MR-U->U                                 "VAL
L3:     U*YL:FR                                 "*S
        NOOP                                    "NOOP
        ML->U
        WRITE U, GOTO OPSEQ                     "store last result
NEG:    '33'D->U
        U-MR->U, IO XOR W->V, GOTO L2           "u=val,v=next pt
NEGL:   '33'D->U                                "2*16.5
        U-MR->U, GOTO L3                        "u=val,negative

"MACRO LTOM255,  DA   DOP=DY->W    D/A output

LTOM255:  W->DA(DI)
        ANALOG->DEV                             "select ANALOG IO
        READ, '16'D->XL                         "get first value
        ABS DX->U, '31'D->J                     "do loop 32 times
        U*SINV2                                 "2*ABS(L)/S
        '33'D->T, U:G                           "2*16.5,no ovf for SCLV
        MR+T->V, '8'D->W, DX->T,                "V=SL, save old L
```

```
        READ                        "fetch new
      SHIFT(SCLV)*V:PP,             "float SL
        W-SCLV->W                   "compute N-6
      XL*W:PP, ABS DX->U            "16*(N-6)
      MR->W, '32'D->YL              "W=flt M,YL=fix shift
LMLP: U*SINV2, T->U                 "2*ABS L/S,orig.val
      '33'D->T, W*YL:FR,            "fix M
        MR->U, U:H                  "test sign,u=16*(N-16)
      MR+T->V, '8'D->W, DX->T,      "V=SCL,save old val
        READ                        "get next
      SHIFT(SCLV)*V:PP, ML+U->V,    "float M
        W-SCLV->W,                  "compute N-6
        IF H<0 GOTO NEGLM           "V=unsigned result
      -1 XOR V->V, XL*W:PP          "form pos. result
      ABS DX->U, MR->W              "W=flt M,u=ABS L
LM2:  V->IO, DEC J, IF J>0 GOTO LMLP "output result
      GOTO OPSEQ
NEGLM: '177'O XOR V->V, XL*W:PP     "form neg result
      ABS DX->U, MR->W, GOTO LM2    "W=flt M,U=ABS L
      END
```



```
      TITLE ASLINE
      EXPAND
      NOLIST
      SYMBOL
      ENTRY LINIT, SNDCMD, SNDCHR, ASINT
      EXT GOTO, OPSEQ
      EQU EA='31'O              "DA + DBLE

"ASYNCHRONOUS LINE INTERFACE COMMANDS
"DEFINITIONS FOR INTERFACE CONTROL REGISTER BITS
      EQU M='77400'O, RESET='100000'O
      EQU IEN='140000'O         "INT ENABLE
      EQU DATA='160000'O        "DEFAULT IS CMD
      EQU RD='170000'O          "READ STATUS/DATA FROM DEV
      EQU WRT='174000'O         "WRITE CMB TO DEV CMD/DATA
      EQU L1='176000'O          "SELECT LINE 1
      EQU LO='177000'O          "SELECT LINE 0
      EQU DEN='177400'O         "ENABLE AP DATA ONTO CMB

"BAUD RATE CODES FOR LINIT
"       0000    50
"       0001    75
"       0010    110
"       0011    134.5
"       0100    150
"       0101    300
"       0110    600
"       0111    1200
"       1000    1800
"       1001    2000
"       1010    2400
"       1011    3600
"       1100    4800
"       1101    7200
"       1110    9600
"       1111    19200

"       MACRO LINIT, ARG     ARG=BAUD RATE CODES FOR L1,LO

LINIT:  '11'O->DEV, V->U, U->T  "MAKE V AVAILABLE
        XB=M+RESET+DEN+LO+L1,
          XB OR W -> V          "V=RESET, BAUD RATE
        19->J                   "COUNT FOR PULSE WIDTH
        V->IO,                  "START RESET PULSE
          V-RESET->V            "PREPARE RESET END
        DEC J,
          IF J>0 GOTO .         "HOLD FOR 5 MIC.SEC.
        V->IO,                  "END RESET
          YB=M+DEN+WRT+L1+LO+'116'O,
          YB->V                 "MODE SETUP
        V->IO,                  "SET MODE
          19->J, 19->W          "SET COUNT
        T:F->U, U->V,           "RESTORE V
          XB=M+DEN+L1+LO+'116'O,
          XB->T                 "SETUP END WRT
        T->IO,                  "END WRT FOR MODE
```

```
          DEC J,
          IF J>0 GOTO .          "WAIT
        XB=M+DEN+WRT+LO+L1+'66'O,
          XB->IO,                "START WRT FOR CMD
          W->J                   "SET UP COUNT
        XB=M+DEN+L1+LO+'66'O,XB->T
        T->IO, DEC J,            "END WRT FOR MODE
          IF J>0 GOTO .          "WAIT
        XB=M+DATA+IEN, XB->T,
          READ                   "FETCH NEXT MACRO
        T->IO, EXEC MACRO        "EXIT TO NEXT MACRO


"       MACRO SNDCHAR,ARG   ARG=LINE CODE, U = CHAR

SNDCHR: V->W, W:H->V,            "SAVE V
          GOTO SNDC

"       MACRO SNDCMD, LINECODE      U = CTRL CHAR

SNDCMD: W - DATA:H -> V,         "FORM CTRLS
          V->W                   "SAVE V
SNDC:   '11'O -> DEV,            "ADDRESS LINE UNITS
          V + U ->V              "ADD COMMAND
        V -> IO,                 "SEND ADDRESS
          YB=DEN+WRT,
          YB + V -> V            "PREPARE WRT
        V->IO,                   "START WRT PULSE
          V - WRT -> V           "PREPARE WRT END
        READ, V->T, W->V         "FETCH NEXT MACRO
        T->IO, EXEC MACRO        "END WRT, EXIT

        EQU RCVMSK='2400'O       "RCVR RDY BITS FOR L1 & LO

"MACRO ASINT, DA(LINE DESCS), A(RCV ROUTINE), A(TR ROUTINE)
"       DOP = DY->DA, DBLE
"       RETURNS V=A(LINE DESCRIPTOR), U=INPUT CHAR AND
"       BRANCHES TO THE TR OR RCV ROUTINE IF TRRDY
"       OR RRDY IS ACTIVE. IF NO LINES READY,
"       CONTINUE IN LINE.
"       LINE DESCRIPTOR FORMAT:
"       0:  MASK FOR TRRDY AND RRDY BITS FOR LINE
"       1:  LINE CODE=M+DATA+Ln+IEN  (IF INTERRUPTS USED)
"       2:  A(NEXT CHARACTER PAIR IN D). CHARACTER COUNT
"       4:  CURRENT CHARACTER, A(OUTPUT COMPLETE ROUTINE)
"       6:  OUTPUT ACTIVE FLAG, A(INPUT ROUTINE)

ASINT:  2->J                     "NUMBER OF LINES
NEXT:   DA->U, 8:H->V            "A(LINE DESC)->U
        DEC J, IF J>0 GOTO LOOP, "BRANCH IF MORE
          READ, INC DA(V)        "FETCH MASK/CODE
        READ(EA)                 "SKIP OVER EXITS
        GOTO OPSEQ               "NO LINES READY

LOOP:   '11'O->DEV, DX->V        "V=TRRDY/RRDY MASK
        IO AND V -> V, RD->W     "TRRDY OR RRDY?
```



```
        RCVMSK AND V             "RRDY?
        IF H=0 GOTO NEXT,        "NEITHER READY FOR LINE
          DY->V
        IF H=0 GOTO TRRDY,       "ONLY TRRDY
          V->IO, V+W:H->V        "ADDRESS LINE
        V->IO, V-W:H->V,         "SET CODE MASK
          '377'O->W              "CHAR CODE MASK
        U->V, V->T,READ(EA)      "V=A(LINE DESC)
        IO AND W -> W            "CLEAN OFF CHAR
        T->IO, W->U              "CHAR -> U, END RD
        DX->W, GO TO GOTO        "EXIT TO RCV

TRRDY:  READ(EA), U->V           "V=A(LINE DESC)
        DY->W, GO TO GOTO        "EXIT TO TR
        END
```

```
        TITLE AVE
        ENTRY AVE
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO AVE,  XYA(DESC)    DESC = C, N
"                               YA, r
" Y(YA+j) = Y(YA+j-1) + C *(Y(YA+j+r)  - Y(YA+j)), j=1,...,N
" Y(0) is not changed

AVE:    Y->J, X*MB:FR, INC XA, INC YA           "set CNT,C
        X->YA, Y->YC                            "Y=Y(YA)
        Y->V, INC YA                            "V=Y(YA)
        MA*Y:FR, INC YA(YC)                     "Y(YA+1)*C
" Do Loop for j=0,...,N
AVEL:   MA*Y:FR, DEC YA(YC+1)                   "Y(YA+j+1+r)*C
        V->Y, INC YA(2), V-ML->V                "new Y(YA+j)
        ML+V->V, MA*Y:FR,                       "new Y(YA+j+1)
                INC YA(YC), DEC J,              "C*Y(YA+j+2)
                IF J>0 GOTO AVEL                "j+1->j
        READ, GOTO EXMAC                        "exit
        END
```



```
        TITLE BLDPOP
        ENTRY BLDPOP
        EXT EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO BLDPOP, CNT, YA(CHNLBLKS)+1, XA(POPTAB)
"       DOP = DY->J, READ
"Assume CHANNEL BLOCKS in Y PAD = MREC, PA, PB, PC
"Output POPTAB in X PAD = PA, PD, PF, PE, PC, PB

BLDPOP: DX->YA, DY->XA, DEC J                   "adjust count
        Y->U, INC YA                            "U=PA
LOOP:   U->X, INC XA, Y+U->U,                   "PA->X,PB->V
                Y:H->V, INC YA                  "PD=PA+PB->U
        U->X, INC XA, Y+U->U,                   "PD->X, PC->W
                Y->W, INC YA                    "PF=PA+PB+PC->U
        U->X, INC XA, W+V->U, INC YA            "PF->X,PE=PB+PC->U
        U->X, INC XA, W->U                      "PE->X, PC->U
        U->X, INC XA                            "PC->X
        V->X, INC XA, Y->U, INC YA,             "PB->X,next PA->U
                DEC J, IF J>0 GOTO LOOP         "DO CNT BLKS
        READ, GOTO EXMAC                                " exit
        END
```

```
        TITLE BLKSHFTS
        ENTRY NORMX, NORMXY, SCALEXY, SCALEX, LDMA,
              MULFRC, MULTFRXC, MULTXC, MULTXYC
        EXT   EXMAC, ONESTEP
        EXPAND
        NOLIST
        SYMBOL

"MACRO NORMX, XYA(DESC)  DOP=10  DESC->YA, XA
"                                    N
"  Computes X*2**U->Y  for N X's, 0<=U<16
"MACRO MULTXC, XYA(DESC)  Computes  MA*X->Y
"MACRO NORMXY, XYA(DESC)  DOP=10  DESC = XYA, N
"  Computes (XY*2**U)->XY  for N values, 0<=U<16
"MACRO MULTXYC, XYA(DESC)  Computes (XY*MA)->XY
"MACRO SCALEXY, XYA(DESC)  DESC = XYA, N
"   Computes (XY*2**U)->XY  for N values -16<=U<0
"MACRO MULFRC, XYA(DESC)  Computes XY*MA->XY
"MACRO SCALEX, XYA(DESC)  DOP=10  DESC = YA,XA
"                                      N
"Computes X*2**U:FR->Y  for N values  -16<=U<0
"  This is twice the inverse of NORMX
"MACRO MULTFRXC, XYA(DESC) Computes X*MA:FR->Y
"MACRO LDMA, ARG  DOP = 0,2,4,6
"  Puts ARG in MA (signed)

NORMX:  SHIFT(U)*MB:PP                  "Set MA=2**U
MULTXC: Y->XA, INC YA, X->U             "Set XA,U=YAVAL
        Y->J, MA*X:P2, INC XA           "Set J
        MA*X:P2, INC XA, DEC J,         "start first mult
                U->YC
        MR->U, MA*X:P2, INC XA,         "U=first result
                YC->YA
        U->Y, INC YA, MR->U,            "store Y,U=Y(+1)
                MA*X:P2, INC XA, DEC J, "start Y(+3)
                IF J>0 GOTO .           "repeat N times
        READ, GOTO EXMAC                "done

"MACRO NORMXY, XYA(DESC)  DOP=10  DESC = XYA, N
" Computes (XY*2**U)->XY  for N values  0<=u<16
"MACRO MULTXYC, XYA(DESC)  Computes (XY*MA)->XY

NORMXY: SHIFT(U)*MB:PP                  "MA*2**MB
MULTXYC: Y->J, X->YA, X->XC             "set J,YA
        MA*Y:PP, INC YA, XC->XA,        "start first RH
                DEC J
        MA*X:P2, INC XA, DEC J          "first LH
        MA*Y:PP, MLMR->UV, DEC YA       "UV=shifted RH
NXYLP:  V->Y, INC YA(2), MR+U->U,       "UV=result
                MA*X:P2, DEC XA         "start lh of result(+1)
        U->X, INC XA(2), MLMR->UV,      "UV=shifted RL of next
                MA*Y:PP, DEC YA,        "start RH of result(+2)
                DEC J, IF J>0 GOTO NXYLP "repeat N times
        READ, GOTO EXMAC

"MACRO SCALEXY, XYA(DESC)  DESC = XYA, N
```



```
"   computes XY*2**U->XY,  for N values -16<=U<0
"MACRO MULFRC, XYA(DESC)  Computes XY*MA->XY

SCALEXY: SHIFT(U)*MB:PP                 "set up MA
MULFRC: X->XC, X->YA, Y->J              "set up YA,J
        XC->XA, DEC J, MA*Y:PP,         "set up XA
                INC YA                  "shift RH
        MA*X:P2, INC XA, 3->YC          "shift LH of first
        0->U, ML:H->V, MA*Y:PP,         "UV=shifted RH
                INC YA
        MLMR+UV:GH->UV, MA*X:P2,        "UV=result
                DEC XA
MFRLP:  U->X, V->W, 0:F->U, ML->V,      "store LH result
                MA*Y:PP, DEC YA(2),     "shift RH(+2)
                INC XA(2)
        W->Y, MLMR+UV:GH->UV,           "store RH result
                INC YA(YC), MA*X:P2,    "shift LH(+2)
                DEC XA, DEC J,          "repeat N times
                IF J>0 GOTO MFRLP
        READ, GOTO EXMAC, INC XA(2)     "leave XA=YA

"MACRO SCALEX, XYA(DESC)  DOP=10  DESC = YA, XA
"                                      N
"  Computes X*2**U:FR->Y for N values -16<=u<0
"MACRO MULTFRXC, XYA(DESC)  Computes X*MA:FR->Y

SCALEX: SHIFT(U)*MB:PP                  "set up MA

MULTFRXC:  Y->XA, X->T, INC YA          "set XA
        Y->J, T->YA                     "set up YA,J
        DEC J, MA*X:FR, INC XA          "start first
        MA*X:FR, INC XA                 "start second
        ML->W, MA*X:FR, INC XA          "W=first result
        W->Y, INC YA, ML->W,            "store result
                MA*X:FR, INC XA,        "repeat N times
                DEC J, IF J>0 GOTO .
        READ, GOTO EXMAC

"MACRO LDMA, ARG  DOP=0,2,4,6  Puts ARG in MA (signed)

LDMA:   DISABLE                         "Will not serve ints
        W*MB:22                         "set up MA
        READ, GOTO ONESTEP              "EXMAC, ENABLE INTS
        END
```

```
        TITLE CODING
        ENTRY ENCODE, DECODE
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"ENCODE, CNT, YA(LIST), XA(A(CODE TABLE))
"DECODE, CNT, YA(A(CODE TABLES)), XA(CODES LIST)

        EQU DI='12'O

ENCODE: DEC J, DX->YA, DY->XA                   "set J,XA,YA
        X->W                                    "first table address
ENLP:   W->DA(DI), INC XA                       "point at table
        READ, 1->W, Y->U                        "U=value
        DX-U:G, READ, -1:H->V                   "init. code
        READ, DX-U:G                            "test second value
        READ, DX-U:G, W+V:H->V,                 "test table(i+2)
              IF G<0 GOTO .                     "repeat if U>table
        V->Y, INC YA, X->W,                     "store code index
              DEC J, IF J>0 GOTO ENLP           "finish list?
        GOTO OPSEQ                              "exit

"DECODE, CNT, YA(A(CODE TABLES)), XA(CODES LIST)

DECODE: DX->YA, DY->XA, DEC J                   "point at data
        X->W, INC XA                            "first code
        Y+W->W, INC YA,                         "W=A(left value)
              MA*'40000'O                       "0.5->MB
        W->DA(DI)                               "DA(left value)
        READ, X->W                              "fetch,W=next code
        DX*MB:FR, Y+W->W, INC YA                "left*0.5
        READ, W->DA                             "fetch rt value
DCLP:   DX*MB:FR, INC XA                        "right*0.5
        ML->U, READ, X->W, DEC XA(2)            "W=code(+2)
        ML+U->U, DX*MB:FR,                      "val=0.5*(lft+rt)
              Y+W->W, INC YA                    "A(left(+2))
        U->X, INC XA(2), READ, W->DA,           "store result
              DEC J, IF J>0 GOTO DCLP           "done?
        GOTO OPSEQ                              "exit
        END
```



```
        TITLE DAOPS
        ENTRY STUDA,LDUDA
        INTRLV
        EXT OPSEQ
        EQU  DA=10

STUDA   W->DA(DA)                               "DESTINATION
        WRITE U, GOTO OPSEQ                     "STORE U IN D(DA)

LDUDA   W->DA(DA)                               "SOURCE ADDRESS
        READ
        DX->U, GOTO OPSEQ                       "FETCH D(DA)->U
        END
```

```
        TITLE DECIM
        ENTRY DECIM, SIZECHK
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"DECIM - FFT DECIMATE ROUTINE
"MACRO DECIM,  D(DESC)  DOP=0,2,4,6
"  Assumes a descriptor:  COUNT, DA(DATA)
"                         SCALE, L
"                             K, RB  in D
"   L,K, and RB are filled in by FFT1
"SUBROUTINE SIZECHK.  Checks V, U=A(DESC)
"   If V>2**13, shifts data right by 1 or 2 and adjusts SCALE

        EQU  DI='12'O

DECIM:  W->DA(DI:D)                     "point at DESC
        READ, 0+U->U                    "fetch N, A(DATA)
        DX->V, DY->DA(DI:D)             "V=COUNT, F=A(DATA)
        SHIFT(SCLV)*4:PP                "2**(16-r)
        0->T, 0->W, V->J                "XYBASE=0, J=COUNT
        T->XC, T->YA, MR->V             "MR=2**(16-r)
        DEC J, XC->XA, DCMV+T->U,
                READ
LOOP:   MR+V:H->V, U->XA, U->YC,        "Inc ptr
                DX->X, ABS XB->U        "next dest, load real
        DY->Y, ABS YB->U, YC->YA,       "load imaginary
                U OR W->W               "size real
        U OR W->W, DCMV+T->U,           "size imaginary
                READ, DEC J,            "decrement ptr
                IF J>0 GOTO LOOP        "fetch next pair
        W->V, GOTO OPSEQ                "V=max data, exit

"SUBROUTINE SIZECHK.  Checks V, U=A(DESC)
"   If V>2**13, shifts data right by 1 or 2 and adjusts SCALE

SIZECHK:  2->W                          "V=SIZE
        SCLV-W->W, 1:G->V               "SCL(SIZE)-2
        U->DA(DI:D), W-V:H->V           "set to read DESC,shift # is 1 more
        IF H<0 GOTO SCALE,              "shift needed
                SHIFT(V)*MB:PP,         "set MA for RS
                CLR XA, CLR YA          "point at data
        READ, RTN
SCALE:  READ, W:H->V, X->XL,            "XL=X(0), get count
                0->U, -2->W             "dec for file
        V->U, DX:H->V, READ (DI),       "V=COUNT
                INC DA(U)
        V->J, MA*XL:FR, INC XA,         "shift X(0)
                DX-U->U                 "new scale
        MA*Y:FR, INC YA, U->T, DEC J    "shift Y(0)
        MA*X:FR, DEC XA, ML->U          "shift X(1), U=new X(0)
SCLP:   U->X, INC XA(2), ML->U,         "store X(j), U=Y(j)
                MA*Y:FR, DEC YA         "shift Y(j+1)
        U->Y, INC YA(2), ML->U,         "store Y(j), U=X(j+1)
```



```
        MA*X:FR, DEC XA,        "shift X(j+2)
        DEC J, IF J>0 GOTO SCLP "j+1->j, loop
WRITE T, INC DA(W),             "store new scl
        CLR XA, CLR YA
READ (DI:D)
RTN
END
```

```
        TITLE DIVBYY
        ENTRY DIVBYY
        EXPAND
        NOLIST
        SYMBOL

"MACRO DIVBYY, YA   DOP=DX->YA
"  Computes U/Y  to 32 bit accuracy in UV
"    0 <= U <= Y

DIVBYY:  '31'D->J, 0->W, 0->V
        U-Y->U, U->T                    "this usually fails
LOOP:   VW+VW+LGCO:GH->VW               "2*VW+(subt OK)
        IF G<0 GOTO Z,                  "subt no good?
                U+U->U                  "double new rem.
NEXT:   DEC J, IF J>0 GOTO LOOP,        "done all bits?
                U-Y->U, U->T            "next trial subt
        1->T                            "wait for result
        IF G>0 GOTO INC, READ           "rem > .5?
        V->U, W->V, EXEC MACRO          "no, exit
INC:    VT+OW:GH->UV, EXEC MACRO        "inc result, exit
Z:      T+T->U, GOTO NEXT               "double rem
        END
```



```
        TITLE DOINT
        ENTRY DOINT
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"CMD MACRO DOINT, INT#  -  push int routine on stk if pending

        EQU  STK='10'0, DI='12'0

DOINT:  W->DEV, W->DA(DI)               "int, vector
        CLR SBIT                        "clear S if set
        IF STAT=0 GOTO OPSEQ, READ      "exit if not set
        WRITE (STK) DX                  "push routine on STK
        GOTO OPSEQ                      "done
        END
```

```
        TITLE DOP
        ENTRY EXMAC
        EXPAND
        NOLIST
        SYMBOL

        EQU DI='12'O, EA='31'O

"These ops start at location 0

DOP0:   DX->W, DX->T                    "D(ARG1)->W,T
DOP1:   DX->W, DX->T, READ              "D(ARG1)->W,T, next word
DOP2:   DY->W, DX->T                    "ARG1->W,T
DOP3:   DX->T, DY->W, READ              "ARG1->W,T, next word
DOP4:   DY->XA, XA->XC,                 "select XA, branch
                IF J>0 PSB=XW, CONT
DOP5:   DY->XA, XA->XC,                 "select XA, branch
                IF J>0 PSB=XWR, CONT
DOP6:   DX->YA, YA->YC,                 "set YA, branch
                IF J>0 PSB=YW, CONT
DOP7:   DX->YA, YA->YC,                 "set YA, branch
                IF J>0 PSB=YWR, CONT
DOP10:  DX->YA, DY->XA                  "set XYA=ARG
DOP11:  DX->YA, DY->XA, READ            "set XYA=ARG, next word
DOP12:  DY->XA                          "set XA
DOP13:  DY->XA, READ                    "set XA, next word
DOP14:  DX->YA                          "set YA
DOP15:  DX->YA, READ                    "set YA, next word
DOP16:  DY->J                           "ARG1->J
DOP17:  DY->J, READ                     "ARG1->J, next word
DOP20:  NOOP
DOP21:  NOOP
DOP22:  DY->DA(DI)                      "ARG1->FILE
DOP23:  DY->DA(DI:D)                    "ARG1->FILE, DBLE
DOP24:  DY+U->W                         "ARG1+U->W
DOP25:  DY+U->W, READ                   "ARG1+U->W, next word
DOP26:  DY+V->W                         "ARG1+V->W
DOP27:  DY+V->W, READ                   "ARG1+V->W, next word
DOP30:  NOOP
DOP31:  NOOP
DOP32:  NOOP
DOP33:  NOOP
DOP34:  NOOP
DOP35:  NOOP

"  PS->D and D->PS routines
"  These can only be reached via DOP or IF J>0 PSB=

DOP36:  DY->W, READ,                    "CNT-1->W
                IF J>0 PSB=RDPS, CONT   "GOTO RDPS
DOP37:  DY->J, READ, CONT               "set CNT-1
LOOP:   YB=DY, READ, PSVWRT             "DY->PSV
        XB=DX, YB=DY, READ, PSLWRT      "DXDY->PSL
        XB=DX, YB=DY, DEC J, PSRWRT     "DXDY->PSR, DEC CNT
        IF J>0 PSB=LOOP, CONT, READ     "DONE?
        EXEC MACRO                      "next macro
```



```
RDPS:   W->J, DX->W, CONT               "ARG1->J
        W->DA(DI:D), CONT               "DEST address, DBLE
RDLP:   PSVRD, WRITE, XB=0              "0,PSV->DLDR
        PSLRD, WRITE, DEC J             "PSL->DLDR
        PSRRD, WRITE, IF J>0 PSB=RDLP   "PSR->DLDR, done?
        READ (EA), CONT                 "fetch next macro
EXMAC:  EXEC MACRO
XW:     X->T, X->W, XC->XA              "X(ARG1)->T,W
XWR:    X->W, X->T, XC->XA, READ        "X(ARG1:->W,T, next word
YW:     Y->W, YC->YA                    "Y(ARG1)->W
YWR:    Y->W, YC->YA, READ              "Y(ARG1)->W, next word
        END
```

```
        TITLE DOTXY
        ENTRY DOTXY, DOTX
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO DOTXY, XYA(DESC)   DOP=LDXYA
"       DESC-> K, N           X, Y
"              BA, FA
"MACRO DOTX,  XYA(DESC)   DOP=LDXYA
"       DESC-> K, N           X, Y
"              BA, FA

DOTXY:  Y->J, INC XA, INC YA,              "set N
                0->U, 0:H->V, 0*0          "MP,U,V=0
        Y->XA, X->YA                       "set FA,BA
LOOP:   MLMR+UV:GH->UV, X*Y,               "accumulate prods
                INC XA, INC YA, DEC J,     "X(FA)*Y(BA)
                IF J>0 GOTO LOOP           "do loop N+1 times
        MLMR+UV:GH->UV, READ,              "add last prod
                GOTO EXMAC                 "exit
DOTX:   Y->J, INC XA, INC YA,              "set N
                0->U, 0:H->V, 0*0          "MP,U,V=0
        Y->XA                              "set FA
DOTXL:  MLMR+UV:GH->UV, X*X, INC XA,       "accumulate prod
                DEC J, IF J>0 GOTO DOTXL   "X(FA)*X(FA)
        MLMR+UV:GH->UV, READ,              "add last prod
                GOTO EXMAC
        END
```



```
        TITLE EXEC
        ENTRY RTN, EXEC, END, HOSTI, RUN, INTRTN
        EXT   GOTO, IN1, CALL, SETP7
        EXPAND
        NOLIST
        SYMBOL

"MACRO EXEC, xxx    DOP=NOOP
"MACRO RTN, -1      DOP=DY->W
"MACRO END, -1      DOP=DY->W
"MACRO RUN, -1      DOP=NOOP
"MACRO HOSTI, PORT  DOP=0,2,4,6
"MACRO INTRIN, -1   DOP=0,2,4  Enable interrupts and RTN

        EQU  CMD='7'0, STK='10'0, EA='31'0, HST='12'0

EXEC:   CMD->DEV                           "select CMD port
        DA->W                              "rest of CMD
        CLR SBIT, GOTO CALL                "save old EA
RTN:    READ(STK), INC DA(W)               "DEC STK PTR
        READ, INC DA(0)                    "fetch address
        DX->W, GOTO GOTO                   "D(STK)->W
END:    READ(STK), INC DA(W)               "back up STK
        READ, INC DA(0)                    "fetch old EA
        DX->W, YB=DA(EA)                   "FA=EA
        W->DA, GOTO IN1                    "restore EA
RUN:    CALL SETP7, ENABLE                 "set up CMD PORT
        DY->W, GOTO RTN                    "resume execution
HOSTI:  HST->DEV                           "address HST
        NOOP                               "wait for status valid
        IF STAT=1 GOTO.                    "wait for ATTN CLR
        W->DEV, READ (EA)                  "set PORT
        INT HOST, EXEC MACRO               "send Interrupt
INTRTN: ENABLE, GOTO RTN
        END
```

```
        TITLE EXTREM
        ENTRY EXTREM
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO EXTREM  XYA(DESC)   DESC = N, R
"                          YA(output), XA(input)
"Searches a list of N+1 points in x for extreme
"values.  For each extreme found, the value of
"the point and its biased index are recorded in Y.
"Neither end point of the list can be an extreme.
"The YA of the next open cell in the list is
"recorded at the beginning of the list.
"If R extremes are found, the search terminates
"and the next macro doublevord is executed.
"If less then R extremes are found, the next
"macro doublevord is skipped.

EXTREM: X->W, Y->J, INC XA, INC YA              "set J=R,W=input count
        X->YA, Y->XA, DEC J                     "list addresses
        1*1, YA->YC, X->U, INC YA,              "make MR=1
                INC XA                          "get X(-1)
        X-U:G                                   "going up or down
        U-X:G                                   "down case
        IF G<0 GOTO DNO, W-MR->W,               "decreasing
                X->U, INC XA                    "U=X(0)
        X-U:G, GOTO UP1                         "X(1)-X(0)
UPO:    IF G<0 GOTO MAX, X-U:G                  "X(j)<x(j-1)?
UP1:    IF H>0 GOTO UPO, W-MR->W,               "dec N, loop?
                U->Y, X->U, INC XA              "X(j)=Y,X(j+1)=U
        YA->YC, YC->YA, READ,                   "save r ptr
                GOTO EXIT                       "skip MACRO
MAX:    XA->XC, INC YA                          "get index
        XC->V, W:H                              "V=index
        V->Y, INC YA, U-X:G,                    "save in list
                DEC J, IF J>0 GOTO DN1          "continue if room
        YA->YC, YC->YA, GOTO EXIT               "no room, done
DNO:    IF G<0 GOTO MIN, U-X:G                  "X(j) > X(j-1)?
DN1:    IF H>0 GOTO DNO, W-MR->W,               "dec N, loop?
                U->Y, X->U, INC XA              "X(j)=Y,X(j+1)=U
        YA->YC, YC->YA, READ,                   "save r ptr
                GOTO EXIT                       "skip MACRO
MIN:    XA->XC, INC YA                          "save index
        XC->V, W:H                              "V=index
        V->Y, INC YA, X-U:G,                    "save index in list
                DEC J, IF J>0 GOTO UP1          "continue if room
        YA->YC, YC->YA                          "no room,save prt
EXIT:   YC->Y, READ, GOTO EXMAC                 "save ptr at front
        END
```



```
        TITLE FADD
        ENTRY FADD
        NOLIST
        EXPAND
        SYMBOL
        EXT FLOAT2

"MICRO SUBROUTINE FADD
"  Assumes first number has its mantissa in UV, scale in W
"  Second number is in Y with its scale in Y(YA), ML in Y(YA+1),
"       MR in Y(YA+2)
"  Returns with Y and YA unchanged

FADD:   Y-W->W, INC YA, Y->U, U->T      "S2-S1->W,S2->U,MH1->T
        XB*Y:PP, XB=1, W+0:H, INC YA    "MB=ML2,test for S1=S2
        SHIFT(W)*MB:PP, 0-W->W,         "shift(ML2),S1-S2->W
          IF H<0 GOTO SY                "branch if S2 smaller
        SHIFT(W)*V:PP, U:H->W,          "shift(ML1), SL2->W
          IF H=0 GOTO NOSH, MR->U       "branch if S1=S2,ML2->U
        IF SHIFT SMALL GOTO SUV         "shift M1<17 places?
          MA*T:P2, Y->V                 "shift(MH1),MH2->v
        IF SHIFT OVF GOTO FLOAT2,       "shift(M1>32 places?
          Y:G->U, U:H->V, V->T,         "U=MH1,V=ML1,T=MH1
          DEC YA(2)
        GASN+T+GCO:F->U,ML+V:G->V,      "add shifted MH1 to M2
          GOTO TUV
SUV:    V+HCO->U, ML-U:H->V, GOTO ADD2  "add shifted ML1 to M2
NOSH:   Y+T+HCO->U, U+V:H->V,           "MH2ML2+MH1ML1->UV
          DEC YA(2), GOTO FLOAT2

SY:     IF SHIFT SMALL GOTO NORM,       "shift MH2
          MA*Y:P2, U+W->W               "recover S1 in W
        IF SHIFT OVF GOTO FLOAT2,       "shift M2>32 places?
          T:G->U, V:H, DEC YA(2)
        GASN+T+GCO:F->U, ML+V:G->V      "add shifted MH2 to M1
TUV:    U:G, V:H, GOTO FLOAT2           "test mantissa for zero
NORM:   T+HCO->U, ML+V:H->V             "add shifted ML2 to M1
ADD2:   MLMR+UV->UV, GOTO FLOAT2,       "add shifted MH
          DEC YA(2)
        END
```

```
        TITLE FCOS
        ENTRY FCOS, FSIN
        EXT RADIAN, AQUAD, BQUAD, CQUAD, DQUAD
        EXPAND
        NOLIST
        SYMBOL

        EQU TWODPI='1772'O

FCOS:   '2'O+W:H->W, 0->T, V->X, INC XA
        U:G, V:H, TWODPI->RA, U->X, DEC XA
        W->Y, W-'36'O:H
        IF GH=0 GOTO ONE
        IF H>0 GOTO TOOBIG
        CALL RADIAN

QTEST:  '00001'O AND U
        '00002'O AND U, T->U
        IF H=0 GOTO EVEN, DEC XA(2), DEC YA(2)
        IF H=0 GOTO CQUAD
        GOTO AQUAD
EVEN:   IF H=0 GOTO BQUAD
        GOTO DQUAD
ONE:    1->W, 0->V                          "COS 0=1
        '40000'O->U
        GOTO FINISH

FSIN:   '2'O+W:H->W, 0->T, V->X, INC XA
        U:G, V:H, TWODPI->RA, U->X, DEC XA
        W->Y, W-'36'O:H
        IF GH=0 GOTO ZERO
        IF H>0 GOTO TOOBIG
        CALL RADIAN
        '3'O+U->U
        GOTO QTEST
ZERO:   '100000'O->W
        GOTO FINISH
TOOBIG: YB->U, YB:H->V, YB->W, DEC YA,
                YB='77777'O
FINISH:  RTN
        END
```



```
        TITLE FDIVS
        ENTRY FDIVS, SFLOAT2
        LISTOBJ
        SYMBOL

"MICRO Subroutine SDIV
"  Computes X/Y  /  U/W ,  U>0
"  Assumes a 128 word table of inverse values:
"       TAB(i) = 128/(i+128)    in ROM
"MICRO Subroutine SFLOAT2
"  Assumes test for U=0 performed in G adder

        EQU  INVTAB='1000'O

FDIVS:  U*'256'D:FR                         "Computes table index
        Y-W->W, XB->T,                      "SCL=SCL(N)-SCL(D)
                XB=INVTAB-'128'D
        ML->V, 1+W->W                       "INC SCL
        '128'D*V:PP, ML+T->V                "shift index back
        V->RA, INVTAB-V:G                   "fetch TVAL, test i=0
        RL*RL:FR, U-MR->U                   "TVAL*TVAL, U=EPS
        U+U->U, 0->T,                       "U=2EPS
                IF G=0 GOTO SS              "D close to 1.0?
        ML+T->T                             "T=TVAL*TVAL
        T*U:FR                              "2E*TVAL*TVAL
        RL->U                               "TVAL
        U-ML->U, 0:H->V, GOTO DIVE          "TVAL-2E*TVAL*TVAL
SS:     IF G=0 GOTO ONE,                    "D=1.0?
                RL-U->U, 0:H->V             "1-2EPS
DIVE:   X*U:FR                              "N*INV(D)
        NOOP
        ML->U, GOTO SFLOAT2                 "result in U, float
ONE:    X->U, 0:H->V, RTN                   "Q=, DONE

"MICRO SUBROUTINE SFLOAT2
"Assumes test for u=0 performed in G adder

SFLOAT2:  ABS U->V, MA*U:P2,                "ABS(result)
                '100000'O->T
        IF G=0 GOTO Z, W-SCLV->W,           "result D
                SHIFT(SCLV)*MB:P2           "adjust scale
        IF SHIFT SMALL GOTO FIN, T->U       "branch if no overflow
        '48'D->U, ML XOR U->V               "correct mantissa sign
        U+W->W, V->U, RTN                   "correct scale, exit
FIN:    MR->U, RTN                          "normalized mantissa
Z:      T->W, 0->U, RTN                     "true zero
        END
```

```
        TITLE FEXP
        ENTRY FEXP
        EXT FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FEXP Floating point exponential
"    Input data is in uv/w
"    Find exp(n) = exp((2**s)*M) = (2**SCL)*MAN
"Uses two tables in ROM:
"    the first is 32 bits = 1/(2ln2), with point at boundary
"    the second is 45 words = EXPTB(n) = exp(n/64), n=0,44

        EQU TBASE='1310'O, LN2='1301'O, INVLN2='1366'O

FEXP:   INVLN2->RA                              "fetch 1/2ln2
        RR*V:PP, 0->T, W-'15'O:H                "s>13?
        RL*MB:PP, U:G, 2+W:H->W                 "shift=s+2
        RR*U:P2, ML->U, W:H,
                IF H>0 GOTO OVERFL              "s too big?
        MLMR + 0U:GH->UV, RL*MB:P2,
                LN2->Y
        MLMR + UV:GH->UV, Y->RA,                "fetch ln2
                GASN+T+GCO:F->T,
                IF H<0 GOTO RSHIFT              "s+2<0
        MLMR+TU:FG->TU, SHIFT(W)*V:PP           "shift v
        MA*U:PP, '77777'O->V                    "shift u
        MA*T:P2, ML+HCO:G->V, MR+V:H
        MLMR+0V:GH->UV, 0->T, XB*XB:PP,
                XB->XL, XB='400'O               "FRAC=.UV
        MLMR+TU:FG->TU                          "2**(S+1)*M/2LN2->T.UV
CONT:   ML+T->V, RL*V:PP,                       "T+1=SCL->V
                '200'O->YL                      "begin ln2*FRAC
        RR*U:PP, RR->V, V:H->W                  "SCL->W
        OML-RLV:GH->UV, XB=TBASE, XB->T,
                RL*MB:PP
        OML+UV:GH->UV, W->Y                     "store scl->Y0
        MLMR+UV:GH->UV, '400'O->W               "uv=ln2*(FRAC-1)
        U*YL:2P, U AND W, '777'O->W             "u*64, round?
        U AND W->V, V->YL, V->W,                "mask 7 lead 0
                XL*XL:PP                        "ER->YL,E=VW
        T-ML:G->U, 0->T,                        "TBASE-(-N)->U
                IF H=0 GOTO EPSL
        U-ML:G->U, V-'1000'O:H->V               "round n+1->u,E-1/64->v

EPSL:   V*YL:2P, V->XL                          "EL*ER, EL->XL
        U->RA, '77777'O->U                      "fetch EXPTB
        MLMR+0U:GH->UV, MA*XL:22,               "EL*EL
                GASN+T+GCO:F->T
        GASN+T+GCO:F->U, ML+U+HCO:G->V,         "2(EL*ER)->UV
                MR+V:H, '52525'O*YL:PP
        MLMR+UV:GH->UV, MA*XL:P2,               "E*E/2->UV
                0->T, 1->YL                     "EL/3
        XBO+TML:GH->UV, V->YL,                  "begin 1+E/3
                XL*YL:PP, XB='100000'O          "(E*E/2)R->YL
```



```
        MLMR+UV:GH->UV, 0->T,                   "(1+E/3)->UV
                IF G<0 GOTO MTBL
        U*YL:PP
MTBL:   MR->U, W:H->V, '77777'O->W              "E->UV
        T+GCO:F->T, ML+V+HCO:G->V,              "E+E*E*(1+E/3)/2->UV
                MR+W:H, Y->W                    "SCL->W
        XBML+UV:GH->UV, 0->T,                   "(1+E)+E*E(1+E/3)/2->UV
                XB='100000'O
        RL+0:H
        RL*V:PP, '77777'O->Y                    "begin EXPTB*(E TERM)
        T:G, U:H, RR*U:PP,
                IF H=0 GOTO FLOAT2              "MANTISSA>0
        MLMR->UV, RL*MB:PP
        MLMR+UV:GH->UV, T+GCO:F->T
        ML+T+GCO:F->U, MR+U+HCO:G->V,
                Y+V:H, XB=0, XB->T
        T:G, U:H, GOTO FLOAT2
RSHIFT: MLMR+TU:FG->TU, '20'O+W:H->V
        SHIFT(W)*U:PP, 0->U                     "shift u
        IF H<0 GOTO RBIG,                       "S+2<-16
                MA*T:P2                         "shift T
        ML->V, 0->T, XB='400'O,
                XB*XB:PP, XB->XL
        MLMR+0V:GH->UV, GASN+T:F->T,
                GOTO CONT
RBIG:   SHIFT(V)*MB:P2, T->U,                   "shift T>16 right
                0->T, '400'O->XL
        IF SHIFT SMALL GOTO ONLYT,              "-32<S+2<-16
                0:H->V, XB=U, XB:G, GASN+T:F->T,
                XL*XL:PP
        '40000'O->U                             "S+2<-32
        ML->W, RTN                              "# -> 1
ONLYT:  GASNML+TO:FG->UV, GOTO CONT
OVERFL: IF G>0 GOTO POSINF                      "M>0?
        0->U, 0:H->V
        '100000'O->W
        RTN
POSINF: YB='77777'O, YB->U, YB:H->V, YB->W      "# -> +inf
DONE:   RTN
        END
```

```
        TITLE FFT1
        ENTRY FFT1
        EXT   OPSEQ, SIZECHK
        EXPAND
        NOLIST
        SYMBOL

"MICRO FFT1,  DA(DESC)   DOP=0,2,4,6

        EQU DI='12'O

FFT1:   W->U, CALL SIZECHK              "shift data if needed
        DX*'100000'O:PP                 "count*0.5
        READ, INC DA(0)                 "fix scale
        WRITE DXYB, YB=1, ML->V         "set SCALE, L=1
        WRITE VYB, YB='512'D            "set K, RB
        V->J, CLR XA, CLR YA            "set lop count
        Y:G->V, INC YA, INC XA,         "V=Y(0)
                '100000'O*MB, 0->W,     "MA=-1.0
                DEC J                   "W=SIZE
L:      Y+V->U, MA*X:FR, V-Y:H->V,      "U=Y(2j)+Y(2j+1)
                DEC YA, DEC XA          "V=Y(2j)-Y(2j+1)
        U->Y, ABS YB->U, INC YA,        "store new Y(2j)
                X->T, INC XA            "T=X(2j)
        V->Y, ABS YB->V, INC YA,        "store new Y(2j+1)
                U OR W->W, ML+T:F->U    "size new Y(2j)
        V OR W->W, U->X,                "size new Y(2j+1)
                ABS XB->U, DEC XA,      "store new X(2j+1)
                T-ML->T                 "T=X(2j)+X(2j+1)
        U OR W->W, T->X,                "size new X(2j+1)
                ABS XB->U, INC XA(2)    "store new X(2j)
        U OR W->W, INC XA, INC YA,      "size new X(2j)
                Y->V, DEC J,            "V=Y(2j+1)
                IF J>0 GOTO L           "j+1->j, loop
        W->V, GOTO OPSEQ                "V=SIZE, EXIT
        END
```



```
        TITLE FFT2
        ENTRY FFT2
        EXT   FFTPASS, SIZECHK, OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO FFT2,  DA(DESC)   DOP=0,2,4,6
"   Repeated step of FFT does FFTPASS log(COUNT)-1 times
"   Upon exit, SCALE word in DESCRIPTOR is updated.
"   The fft result is in DATA PAD starting at 0.

        EQU  EA='31'O, DI='12'O

FFT2:   W->U, CALL SIZECHK              "size data, U=A(DESC)
        READ, '100000'O->XL             "XL=0.5
        DX->T, DY->V, READ              "SCALE, L
        XL*DX:PP, V+V->V                "V=new L
        MA*DY:PP, V->XC, V->YA          "L->YA,XC
        ML->U, READ, INC DA(-4),        "U=new K
                XC->XA, YA->YC          "XA=YC=L
        WRITE TV, ML->V                 "update L, V=new RB
        WRITE UV, IF G=0 GOTO OPSEQ     "store new K,RB
        YC->J, U-1->U                   "set CNT=L,U=K-1
        U->DA('14'O)                    "save in F(14)
        V->RC, 0->V, 0->W,              "set RC
                IF G=0 GOTO LAST        "last pass?
LP:     V->RA, CALL FFTPASS             "RA=0, do group
        DA('14'O)->U, INC XA(XC),       "get k
                INC YA(YC-1)            "XYA(next group)
        U-1->U, YC-J                    "dec K, J=L
        U->DA, 0->V,                    "New K->F(14)
                IF G>0 GOTO LP          "do next group
        DA(EA)->V                       "get EA
        -2+V->V                         "back up
        V->DA, W:H->V, GOTO OPSEQ       "set EA to repeat
LAST:   V->RA, CALL FFTPASS             "RA=0, do last pass
        W->V, GOTO OPSEQ                "V=size. done
        END
```

```
        TITLE FFTPASS
        ENTRY FFTPASS
        EXPAND
        NOLIST
        SYMBOL

"SUBROUTINE FFTPASS
"  W = SIZE DATA
"  XA = A(X2), XC=L, YA=A(Y2), YC=L
"  J=L, RA=0, RC=512/L
"Assumes a 512 doubleword table in ROM of roots of unity:
"       RL(i) = -COS(2pi*i/1024)
"       RR(i) = -SIN(2pi*i/1024)

FFTPASS: RL*X:FR, '77777'O->YL,          "-COS*X2
         DEC XA(XC)                      "XA->X1
         X*YL:FR, O->T, INC XA(XC),      "X1*1.0
         DEC J                           "XA->X2, J=L-1
         RR*Y:FR, T-ML->T, Y->YL,        "-SIN*Y2
         '77777'O->XL                    "T=COS*X2, YL=X2
         RR*X:FR, ML+T->T,               "-SIN*X2
         ML-T->U, INC YA                 "T=X1+COS*X2,U=X1-COS*X2
LOOP:    T-ML->U, ML+U:H->V,             "U=SIN*Y2+COS*X2+X1
         O->T, RL*YL:FR,                 "V=X1-COS*X2-SIN*Y2
         Y->YL, DEC YA(YC+1),            "-COS*Y2+
         DEC XA(XC)                      "YL=Y2, YA->Y1, XA->X1
         U->X, ABS U->U, INC XA(XC),     "store X1, XA->X2
         ML+T->T, INC RA,                "T=-SIN*X2
         XL*Y:FR                         "1.0*Y1
         V->X, ABS V->V, INC XA,         "store X2, XA->X2+
         U OR W->W, T-ML->T              "T=COS*Y2-SIN*X2
         V OR W->W, ML+T:F->U,           "size new X2
         ML-T->V, RL*X:FR,               "U=Y1+COS*Y2-SIN*X2
         DEC XA(XC)                      "V=Y1-COS+SIN*X2-COS*X2+
         U->Y, ABS U->U, INC YA(YC),     "store Y1
         XL*X:FR, O->T,                  "1*X1+
         INC XA(XC)
         V->Y, ABS V->V, INC YA(2),      "store new Y2
         U OR W->W, T-ML->T,             "size new Y1
         RR*YL:FR                        "T=COS*X2+,-SIN*Y2+
         V OR W->W, ML+T->T,             "size new Y2
         ML-T->U, RR*X:FR,               "T=X1+COS*X2
         DEC J, IF J>0 GOTO LOOP         "U=X1-COS*X2-SIN*X2+
         RTN
         END
```



```
        TITLE FFTRL
        ENTRY FFTRL
        EXT   OPSEQ, SIZECHK
        EXPAND
        NOLIST
        SYMBOL

        EQU  DI='12'O

"MACRO FFTRL,  DA(DESC)  DOP=0,2,4,6
"  FFTRL is the final pass of a Real FFT.  To perform a
"  Real FFT of order 2N on Xi, i=0,...,2N-1, first do a Complex
"  FFT of order N on Zj, Rl(Zj)=X(2j) and IM(Zj)=X(2j+1),
"  j=0,...N-1.   The final RLFFT pass computes the first N+1
"  bins of the complex spectrum using the formulae:
"       Let Z(j)=X(j)+iY(j)
"       Z(0)'=X(0)+Y(0), Z(N)'=X(0)-Y(0)
"       For j=1,...,N/2:
"          x1=X(j)+X(N-j), x2=Y(j)+Y(N-j)
"          y1=Y(j)-Y(N-j), y2=X(N-j)-X(j)
"          X(j)'   = x1+SIN(t)*y2 + COS(t)*x2
"          X(N-j)' = x1-SIN(t)*y2 - COS(t)*x2
"          Y(j)'   = y1-SIN(t)*x2 + COS(t)*y2
"          Y(N-j)' = -y1-SIN(t)*x2 + COS(t)*y2
"       The SCALE is decreased by one.

FFTRL:   W->U, CALL SIZECHK              "U=A(DESC), size data
         '100000'O->YL                   "YL=-1.0 OR .5
         Y->U, DX->XC, DX->YA, READ      "XC=YA=N, U=Y(0)
         DX->T, DY->J, READ, YA->YC,     "T=SCALE,
              CLR YA                     "L=N/2->J
         DY*YL:PP, U+U->U,               "RB*0.5
              O:H->V                     "X2=2Y(0)->U
         READ, INC DA(-4),               "F=DA(SCALE)
              GASN+T->T, -4+U:G,         "DEC SCALE
              V->XL                      "XL=0
         O->RA, T->V, X->T,              "init RA, T=X(0)
              YA->YC, YC->YA,            "YA=N, YC=0, W=RC
              ML->W
         WRITE V                         "update SCALE
         W->RC, T+T->V                   "set RC, V=X1
         V->T, Y->V, XL*XL               "T=X1, V=Y, O*O
         RL*U:FR, GOTO START             "-COS*x2
LOOP:    X-T->U, Y:H->V, X->T,           "U=Y2, V=Y1
              YA->YC, YC->YA             "T=X2, YA=N-K
         RR*U:FR, U->XL,                 "-SIN*y2, XL=Y2
              Y+V->U, T-ML->T,           "U=x2, T=x1
              XA->XC, XC->XA, INC YA     "XA=K, YA=N-K+1
         RL*U:FR, W->Y, DEC YA           "-COS*x2, store Y2'
START:   RR*MB:FR, V-Y:H->V,             "-SIN*x2, V=y1
              DEC YA, T-ML->T,           "x1+SIN*y2
              ML+T->U                    "x1-SIN*y2
         RL*XL:FR, YA->YC, YC->YA,       "-COS*y2, YA=N-K
              T-ML->T, ML+U->U           "T=X1', U=X2'
         ML+V->V, ML-V->W,               "y1-SIN*x2,-y1-SIN*x2
              T->X, INC XA               "store X1'
```

```
        V-ML->V, W-ML->W, X->T,          "V=Y1', W=Y2'
                XA->XC, XC->XA,          "T=x1+, -1.0*x1+
                X*YL:FR
        U->X, DEC XA, V->Y, INC YA,      "store X2',Y1'
                INC RA, DEC J,
                IF J>0 GOTO LOOP
        GOTO OPSEQ                       "exit, K=N-K
        END
```



```
        TITLE FILTER
        ENTRY FILTER
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO FILTER,  XYA(DESC)   DESC = N, XA(SOURCE)
"                                  -, M
"  Also have filter coefficients and filter memory
"  at xya(desc)-1 to -m stored as:
"    XYA-1      KM  , BM-1
"    XYA-2      KM-1, BM-2
"               ..., ...
"    XYA-M      K1  , B0
"  N and XA are destroyed, the Bj are updated.
"  The source is replaced by the filter output.
"  F=SRCEj
"  for m=M-1->0,  F=F-Km+1*Bm
"                 Bm+1=Bm+Km+1*F
"  Output = F,  B0 = F

FILTER: Y->XA, 1->V                      "point at source
NEXT:   X->T, XA->XC, DY->YC, DY->XA     "T=SRCE,XC=PTR
        X-V:H->V, U->Y, YC->YA           "DEC N store B1
        V->X, DEC XA, DEC YA, 3->YC      "New N
        IF H<0 GOTO OPSEQ, X*Y:FR,       "done?
                X->XL, Y->W, DEC XA,     "KM*BMM-1
                INC YA(2)
        Y->J, DEC YA(YC)                 "set J=M,Y=BM-2
        T-ML->T, W:H->V, DEC J,          "SRC-KB,V=BM-1
                X*Y:FR, Y->W, DEC XA,    "KM-1*BM-2
                INC YA
        XL*T:FR, DEC J                   "KM*F
LOOP:   T-ML->T, DEC YA(2),              "F=F-KM+1*BM
                X*MB, INC XA             "set MA=KM-1
        ML+V->U, W:H->V, X->XL,          "BM+1=BM+KM+1*F
                DEC XA(2), MA*Y:FR,      "KM-1*BM-2
                Y->W, INC YA(YC)
        U->Y, DEC YA(2), XL*T:FR,        "store BM+1
                DEC J, IF J>0 GOTO LOOP  "KM*F, M-1->M
        T->U, XC->XA, 1->W               "T=U=F
        U->Y, INC YA, ML+V->U,           "store B0,U=B1
                W:H->V, T->X, INC XA,    "store F in output
                GOTO NEXT
        END
```

```
        TITLE FINV
        ENTRY FINV
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FINV
"  Computes INV(UV/W),  ABS (U) >= .5
"  Assumes a 128 word table of inverse values:
"       TAB(i) = 128/(i+128), only 16 bit values are needed

"Algorithm for INVERSE(D)
"  Let D=D1/256 + EPS  such that
"     128 <=D1 <= 256  and -1/512 <= EPS <= 1/512
"  Look up TVAL(D1-128) = 128/D1    (16 bit accuracy)
"  Compute .5ID1 = TVAL-2EPS*TVAL*TVAL  (16 bit accuracy)
"  Then D*ID1 = 1+E  and ABS (E) < 2**-16.
"  Compute .5ID2 = .5ID1*(2-ID1*D),  (16x32 bit multiplies)
"  The result, ID2, satisfies D*ID2 = 1-E**2
"  The output scale is W' = 1-W.

        EQU  INVTAB='1000'O                     "IN ROM

FINV:   U*'256'D:FR, U->T, U->XL        "compute table index
                U:G                     "test for negative
        W->YL, V->W,                    "save scale in YL
                XB=INVTAB-'128'D,       "save D in TW
                XB->V
        IF G<0 GOTO NEG, ML->U          "D<0?,U=table index
        '128'D*U, ML+V:H->V,            "shift index back
                T:G                     "V is RA(TVAL)
        V->RA, 1->V                     "fetch TVAL
        RL*RL:FR, IF G=0 GOTO OF        "TVAL*TVAL, D=0?
COM:    T-MR->U, 0->T, V*MB             "UW=EPS,make true TVAL
        UW+UW:GH->UV, ML+T->T           "UV=2EPS,T=TVAL*TVAL
        T:G, T*U:FR                     "2EPS*TVAL*TVAL
        '40000'O->T, U:H->V, MR->U      "T=.5, U=TVAL
        IF G>0 GOTO DIVE,               "skip if TVAL \=-1
                U-ML->U, W:H            ".5*ID1=TVAL-2ETVSQ
        '77777'O-V->U, W:H              ".5*ID1=1-2EPS
DIVE:   U*W:2P                          ".5*ID1*DR
        MA*XL:22, IF H=0 GOTO Z         ".5*ID1*DL
        00-MLMR:GH->UV,                 ".5-.25DR*ID1
                T-GASN-1+GCO:F->T
C:      TU-MLMR:GH->UV, MA*V:2P,        ".25*(2-D*ID1)
                0->T, '177777'O->W      "now *ID1, rounding bits
        MA*V:2P, '7777'O->V             "do middle
        MA*U:22, T:F->U,                "finally top
                MLMR+VW:GH->VW,         "bottom part+round
                '8'D->XL                "shift amount
        MLMR+UV:GH->UV,                 "add middle
                GASN+T+GCO:F->T,        "carry in T
                1*YL, 1->W              "fetch SCALE
        MLMR+TU:FG->TU, XL*V:PP,        "have result/8,start shift
                IF G<0 GOTO CORRECT     "need a fix?
        MA*U:PP, T-'10000'O:G,          "shift middle part
```



```
                W-MR->W                 "check for 1.0,1-SCALE
FIN:    MA*T:P2, ML:H->V, 0->U, T->X    "shift top
        MLMR+UV:GH->UV,                 "jump if not 1.0
                IF G<0 GOTO EXIT
        '140000'O->U                    "will halve mantissa
        1+W->W, 0->V                    "inc SCL to compensate
EXIT:   MR+U->U, RTN                    "get top, exit
NEG:    '128'D*U, V-ML->V               "shift index back,RA(-TVAL)
        V->RA, -1->V                    "fetch -TVAL
        RL*RL:FR, RL:G,                 "compute TVAL*TVAL
                GOTO COM                "test for TVAL=-1
Z:      0->U, 0->V, GOTO C              "TUV=.5
CORRECT: TU-GASNW:FG->TU,               "DEC TU
                W-MR->W                 "1-SCALE
        MA*U:PP, T:G,                   "shift middle part
                GOTO FIN                "set G NEG
OF:     0->V, U-V->W, RTN               "make spacial 0
        END
```

```
        TITLE FLOAT
        ENTRY FLOAT, FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FLOAT,  UV = INTEGER, result scale in W
"MICRO SUBROUTINE FLOAT2,  UV/W  an UNNORMALIZED NUMBER
"  Must have tested mantissa for 0

FLOAT:  '31'D->W, U:G, V:H              "Scale for int,test val
FLOAT2: ABS U->V, MA*V:PP               "ABS(U)->V, save V
        IF GH=0 GOTO Z, W-SCLV->W,      "value=0
                SHIFT(SCLV)*MB:PP       "adjust scale
        IF SHIFT SMALL GOTO FIN,        "float 0-15 places
                MA*U:P2                 "shift ML
        IF SHIFT OVFL GOTO BIG,         "overflow?
                MLMR->UV, 0->T          "shifted MR
        MR+U->U, '16'D+W->W             "shifted 16+ places
        U:G, V:H, GOTO FLOAT2           "float more
FIN:    MLMR->UV, GOTO EXIT             "least part
Z:      0->U, 0:H->V, '100000'O->W      "make FP0
EXIT:   MR+U->U, RTN                    "Normalized ML
BIG:    MLMR+TU:FG->UV,                 "down shifted mantissa
                '48'D+W->W              "correct scale
        '100000'O XOR U->W, W->U        "invert sign bit
        U:H->W, W->U, RTN                       "put in place
        END
```



```
        TITLE FLOG
        ENTRY FLOG
        EXT FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FLOG Floating point Logarithm
"   Input data is in UV/W
"Uses two tables in ROM:  the first is four words:
"       2*256, ---
"     2.5*256, RA(ln(2))
"       3*256, RA(ln(2.5))
"       ---  , RA(ln(3))
"   the second is 67 words = ln((n+192)/128),  n=0,64
"     ln(2.5), ln(3)

        EQU THRFRTH='60000'O, TWOTHRDS='52525'O, INVTAB='1000'O,
            PTRTAB='1304'O, LOGTAB='1201'O, LN2L='54271'O,
            LN2R='5774'O

FLOG:   PTRTAB->RA                              "first multiplier
        1->RC
        U:G, XB->T, XB=THRFRTH,                 "test for M>0
                W->YL, 0->W                     "save SCL in YL
A:      U-T:G, XB->T, XB=TWOTHRDS               "M:.75
        U-T:G, U:H, IF G>0 GOTO T1              "M:2/3,M<0,M>0?
        WW-UV:GH->UV, XB->T, XB=THRFRTH         "negate M
        IF H<0 GOTO A, T:G                      "M/=0?
        0-T->U, T->W, GOTO FLOAT2               "float lg neg log
T1:     IF G<0 GOTO T1, T:G,                    "find segment
                RL*V:PP, INC RA                 "  start C*MR
        MA*U:PP, RR->YC                         "start C*ML
        MLMR->UW, XB->T,                        "C*MR->UW
                XB=INVTAB-'128'D                "table base
        MLMR+TU:GH->UV                          "U=RR(TINV)
        U->RA, 0->T, XB+U->U,                   "fetch TINV
                XB=LOGTAB-INVTAB-'64'D          "U=RA(TLOG)
        RL*W:PP
        RR*V:PP, '77777'O->W, 0->T              "TINVR*EPSL
        RL*MB:PP, MLMR+0W:GH->VW                "TINVL*EPSL
        T+GCO:F->U, U->RA,                      "fetch TLOG
                MLMR+VW:GH->VW
        MLMR+UV:GH->UV                          "UV=128D
        UV+UV:GH->UV                            "UV=256D
  "Compute D*D/2
        U*V:PP, U->T, V->U                      "DL*DR,TU=256D
        MA*T:PP, '37777'O->W, 0->V              "DL*DL
        OML+VW:GH->VW                           "cross term
        VW+VW:GH->VW                            "double it
        ML+V+HCO:G->V, MR+W:H,                  "V=D*D/2
                '77777'O->W                     "RND for D**3
  "Compute 64D**3, D, and D**3/3
        T*V:PP                                  "D*D**2/2
        MA*'256'D:PP                            "shift DL
        U*MB:PP, 0->T, RR->W,                   "shift DR
```

```
        ML+HCO:G->U, MR+W:H          "U=64D**3
MLMR-TV:GH->UV,                      "DL-D**2/2
        U*'253'O:FR                  "64D**3/192
OML+UV:GH->UV, RL*1:PP               "UV=D-D**2/2
OML+UV:GH->UV,                       "D-D**2/2+D**3/3
        LN2R*YL, O->T                "start LN(2)R*S
MRV+UW:GH->VW, YC->RA                "add TLOG,fetchLN(2)
RL-V->V, LN2L->XL                    "sub L2 from it
VRR-OW:GH->VW,                       "L3=LN(C)-L(2)
        XL*MB                        "start LN(2)*S
MLMR-VW:GH->VW,                      "S*LN(2)R-LN(C)
        GASN-T-1+GCO:F->U
MLMR+UV:GH->UV                       "FLOG*2**31
MA*V:PP, ABS U->V
SHIFT(SCLV)*MB:PP, W:H->V,           "start shift
        O->T, SCLV->W                "DSCL->W
IF SHIFT SMALL GOTO SM,              "shift <=15?
        MA*V:PP                      "shift bottom
MA*U:P2, 1->W,                       "SCL=1
        MLMR->UV                     "middle shifted
OML+UV:GH->UV                        "bottom shifted
MR+U+HCO:G->U, V:H, GOTO FLOAT2      "top shifted
SM:  MA*U:P2, ML+T:F->U,             "shift top
        MR->V, '16'D-W->W            "middle shifted
OML+UV:GH->UV                        "bottom shifted
MR+U->U, RTN                         "top shifted,exit
END
```



```
        TITLE FLTS
        ENTRY FADDS, FSUBTS, FMULTS
        EXT SFLOAT2
        EXPAND
        NOLIST
        SYMBOL

"Assume OP1 mantissa in U, SCL in W;
"       OP2 mantissa in X, SCL in Y

"MICRO SUBROUTINE FADDS
FADDS:  Y-W:H->V, O->T               "SCL2-SCL1
        SHIFT(V)*X:P2, T-V:G->V      "start OP2 shift
        IF H>O GOTO SUV, T-V:G,      "branch if SCL2 biggger
               SHIFT(V)*U:P2         "start OP1 shift
SUM:    IF G>O GOTO SFLOAT2, U:H->V, "shift ok?
               ML+U->U               "add mantissas
        X+V->U, GOTO SFLOAT2         "add unshifted mantissas
SUV:    Y->W, X->U, GOTO SUM         "will shift OP1

"MICRO SUBROUTINE FSUBTS
FSUBTS: Y-W:H->V, O->T               "SCL2-SCL1
        SHIFT(V)*X:P2, T-V->V        "start OP2 shift
        IF H>O GOTO SSUV,            "branch if SCL2 bigger
               SHIFT(V)*U:P2         "start OP1 shift
        IF G>O GOTO SFLOAT2, U:H->V, "shift ok?
               ML-U->U               "shift (man2)-man1
        X-V->U, GOTO SFLOAT2         "subt unshifted mantissas
SSUV:   Y->W, X->U                   "will shift OP1
        U-ML->U, GOTO SFLOAT2        "MAN2-SHIFT(MAN1)

"MICRO SUBROUTINE FMULTS
FMULTS: X*U:FR                       "man1*man2
        Y+W->W                       "SCL1+SCL2->SCL
        ML->U, GOTO SFLOAT2          "result ->U
        END
```

```
        TITLE FLTSOPS
        ENTRY SADD, SSUBT, SMULT
        EXT   FADDS, FSUBTS, FMULTS, EXMAC
        EXPAND
        NOLIST
        SYMBOL

"Short Floating Point Operations.  All operations assume
"  the first operand is in U/V,  U=Mantissa, V=Scale.
"  The second operand is in XY at the location given
"  by the argument.  X=Mantissa, Y=Scale

"MACRO SADD, ARG   DOP=0,2,4   OP2 at XY(ARG)
"  Computes X/Y + U/V -> U/V

SADD:   T->YA, W->XA, V->W,             "XYA(OP2)
                CALL FADDS              "do FADD
        W->V, READ, GOTO EXMAC          "exit

"MACRO SSUBT, ARG   DOP=0,2,4   OP2 at XY(ARG)
"  Computes X/Y - U/V -> U/V
SSUBT:  T->YA, W->XA, V->W,             "XYA(OP2)
                CALL FSUBTS             "do FSUBT
        W->V, READ, GOTO EXMAC          "exit

"MACRO SMULT, ARG   DOP=0,2,4   OP2 at XY(ARG)
"  Computes X/Y * U/V -> U/V
SMULT:  T->YA, W->XA, V->W,             "XYA(OP2)
                CALL FMULTS             "do FMULT
        W->V, READ, GOTO EXMAC          "exit
        END
```



```
        TITLE FMULT
        ENTRY FMULT
        EXT FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FMULT     ML1,MR1 in UV, SCL1 in W
"                       ML2,MR2 in XY(XYA-1), SCL2 in Y(XYA)

FMULT:  Y+W->W, DEC XA, DEC YA          "SCL1+SCL2
        X*V:2P, 0->T                    "ML2*MR1
        Y*U:P2, 0->U                    "MR2*ML1
        X*MB:22, INC XA, INC YA,        "ML2*ML1
                GASNML+TU:FG->TU,       "fix SCL for FRAC MULT
                1+W->W
        GASNML+TU:FG->UV                "add MR2*ML1->UV
        MLMR+UV:GH->UV,                 "add ML1*MR1, float
                GOTO FLOAT2
        END
```

```
        TITLE FSQRT
        ENTRY FSQRT
        EXT FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FSQRT Floating Square Root
"   Input data is in UV/W
"Uses two tables in ROM:
"   the first is 128 words = SQRTAB(n)=srt(n/256), n=128,255
"   the second is 128 words = INVTAB(n)=128/n, n=128,255
"If M is negative, zero is returned in UV, and the original
"   scale in W

        EQU   SQBASE='1367'O, IBASE='1000'O, N128='200'O

FSQRT:  U*'1000'O:2P, U:G                       "move u 9 left
        V->U, W->Y, '177'O AND U->V             "mask 9 lead 0
        IF G>0 GOTO CONT, ML->V, V->T           "M<0?
        0->U, 0:H->V, RTN                       "M<=0,0->UV
CONT:   0->T, ML->W                             "N->W
        YB+W->W, YB=IBASE-N128                  "E=UV
        W->RA, XB+W->W,                         "fetch ITAB
                XB=SQBASE-IBASE
        RR*U:PP                                 "begin ITAB*E=D
        RL*V:PP, '77777'O->V
        T+GCO:F->T, MLMR+OV:GH->UV,
                MA*U:PP
        OML+TU+HCO:FG->TU, MR+V:H,
                W->RA                           "fetch SQBASE
        MLMR+TU:FG->UV, 0->W,                   "D/2->UV
                '100000'O->T
        UV+UV:GH->UV                            "double it
        TW-UV:GH->UV, U->X,                     "(1-D)->UV
                V->Y, U->XL,                    "DL->X,XL,DR->Y
                U*V:PP                          "DL*DR
        0->T, '77777'O->W
        MLMR+OW:GH->UW, XL*XL:PP,               "(1-D)L->XL
                U->XL, V->YL                    "(1-D)R->YL
        OML+TU+HCO:FG->TU, MR+W:H               "2*DL*DR->TU
        MLMR+TU:FG->UV                          "D*D/2->UV
        U*YL:PP, '-40000'O->T                   "(D*D/2)*(1-D)
        XL*V:PP, '77777'O->W
        MLMR+OW:GH->UV, MA*U:PP
        OML+TU+HCO:FG->TU, MR+V:H
        MLMR+TU:FG->UV, 0->W,                   "D*D/2*(1-D)/2-1/2
                '100000'O->T                    "  ->UV
        UV+UV:GH->UV                            "D*D/2*(1-D)-1
        XY-UV:GH->UV, DEC YA                    "1+D-D*D/2*(1-D)
        RR*U:PP, 0->T, '17777'O->W              "SQRTB*(D term)
        RL*V:PP, 4->XL
        MLMR+OW:GH->UV, MA*U:PP
        T+GCO:F->T, MLMR+UV:GH->UV,
                '100000'O*Y:P2, Y->W            "begin S/2
        MLMR+TU:FG->TV, XL*V:PP,
```



```
                '00001'O AND W:H                "test S odd
        ML->W, SQBASE->RA                       "INT(S/2)->W
        TV+TV:GH->UV, 0->T,                     "SQRTB*(D term)
                IF H>0 GOTO SODD
        ML+V:H->V, U:G, GOTO FLOAT2
SODD:   ML+V->V, 1+W->W, RR*U:PP                "srt(1/2)*srt(M)
        RL*V:PP
        MLMR->UV, MA*U:PP, 0->T
        OML+TU+HCO:FG->TU, MR+V:H
        MLMR+TU:FG->UV
        UV+UV:GH->UV, GOTO FLOAT2
        END
```

```
        TITLE FSUB
        ENTRY FSUB
        EXT FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"MICRO SUBROUTINE FSUB
"  Assume one operand in U,V,W = ML,MR,SCL
"  Second operand in XY PAD, current XYA-> --,SCL
"                                 XYA-1-> ML, MR
"  Computes Second operand - First operand

FSUB:   Y-W->W, DEC XA, DEC YA,          "SCL2-SCL1->W
                1*Y                      "MP will be SCL2
        Y->YL, 0->T                      "YL=MR2, wait for test
        IF H>0 GOTO SUV, T-W->W,         "SCL2>?
                SHIFT(W)*YL:PP           "SCALE MR2
        IF SHIFT SMALL GOTO NORM,        "shift 0-16 places?
                MA*X:P2, MR+W->W         "W=SCL1
        IF SHIFT OVFL GOTO FLOAT2,       "shift>32 places
                U:G, V:H, U->T,          "test for 0
                INC XA, INC YA
        GASNML-TV:FG->UV,                "subt from shifted MPL2
                V:H, GOTO FLOAT2         "17-31 place case
NORM:   IF H=0 GOTO NOSH,                "no shift needed?
                TML-UV:GH->UV            "subt from shifted MR2
        MLMR+UV:GH->UV, INC XA,          "add shifted ML
                INC YA, GOTO FLOAT2
NOSH:   XY-UV:GH->UV, INC XA,            "do subt
                INC YA, GOTO FLOAT2

"  SCL2 > SCL1, will shift MLR1
SUV:    SHIFT(W)*V:PP, MR->W             "shift MPR1, W=SCL2
        IF SHIFT SMALL GOTO NS,          "shift ARG1 1-16
                MA*U:P2, Y->V            "shift ML1
        IF SHIFT OVFL GOTO FLOAT2,       "shift >32 places?
                X->U, V:H, X->T,
                INC XA, INC YA
        TU-GASNML:FG->UV,                "MLR1-ML1
                V:H, GOTO FLOAT2         "shift was 16-31 places
NS:     XV-OML:GH->UV                    "MPLR2-MPR1
        UV-MLMR:GH->UV, INC XA,          "subt shifted MPL1
                INC YA, GOTO FLOAT2
        END
```



```
        TITLE INITCHK
        ENTRY INIT, IN1, CMND, SETP7
        EXT EXMAC, GOTO
        EXPAND
        NOLIST
        SYMBOL

" INIT is the initialization routine to be burned into ROM for CHI-5

        EQU     CMD='7'O,                "CMD PORT
                CMNBS='30'O,             "DA of CMD buffer
                DI='32'O,                "FA scratch, DBLE
                STK=0,  "(really 10)     "FA MACRO STK file cell
                EA='31'O,                "FA(execution addr),DBLE
                STKBS='12'O,             "DA of Macro stack
                HST='12'O,               "DEV for testing ATTNQ
                DIN=0,                   "PORT 0
                DOUT=1,                  "PORT 1
                PNLSW='17'O,             "DEVICE 17=PNL SW
                MACBASE='140000'O        "BASE OF ROM MACRO ROUTINE
                                         "must be at '370'O

CHECK:  PNLSW->DEV                       "Address SW
        MACBASE->W                       "entry point
        IF STAT=0 GOTO IN1               "HOST INIT
        GOTO GOTO, ENABLE                "start
        LOC CHECK+'10'O
INIT:   GOTO START, 0->V, CLR S          "set PSA
START:  V->DA(DIN)                       "set PORT 0
        V->DA(DOUT)                      "set PORT 1
        STKBS->W
        W->DA(STK), GOTO CHECK           "init STK
IN1:    CALL SETP7                       "set up CMD port
BRSC:   IF STAT=0 GOTO BRSC              "wait for CMD
CMND:   CMNBS->W, DISABLE
        W->DA(DI)                        "point at CMD buffer
        READ, GOTO EXMAC                 "fetch first MACRO
SETP7:  HST->DEV                         "select HST STATUS
        CMNBS->W
        W->DA(CMD)                       "set up CMD port
        IF STAT=1 GOTO .                 "wait for ATTN clear
        CMD->DEV                         "DEV=CMD PORT
        INT HOST, RTN                    "Interrupt HOST
        END
```

```
        TITLE INTSRV
        ENTRY INTSRV, CALL, GOTO, OPSEQ
        EXT   CMND
        EXPAND
        NOLIST
        SYMBOL

"INTSRV -- INTERRUPT HANDLER, CALL, GOTO

        EQU  CMD='7'0, HST='12'0, STK='10'0, EA='31'0, DI='12'0

INTSRV: GOTO FLIH, DA->W                "set PSA, get EA
FLIH:   CMD->DEV, DISABLE               "CMD first
        W-2->W                          "back up EA
        W->DA, 0->W,                    "Restore file
                IF STAT=1 GOTO CMND     "CMD INT?
        1->T
        W->DEV, W->DA(DI), T+W->W       "DEV=0, DA=0, W=1
INTLOOP: CLR SBIT, W->DEV, T+W->W       "0->S(j),DEV=j+1,W=j+1
        IF STAT=0 GOTO INTLOOP, READ    "WAS S(j) 1? D(j)->DX
        DX->W, GOTO CALL                "W=EA(INT routine)

"MACRO CALL, EA  DOP is DY->W or D(DX)->W for indirect

CALL:   DA(EA)->U, U->T                 "save old EA,U
        WRITE (STK) U, T->U             "EA->STK

"MACRO GOTO, EA  DOP is DY->W or D(DX)->W for indirect

GOTO:   W->DA(EA)                       "set new EA
OPSEQ:  READ(EA)                        "fetch next MACRO
        EXEC MACRO                      "do it
        END
```



```
        TITLE LATRED
        ENTRY LATRED
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO LATRED,  XYA(DESC)     DESC =  K, N
"                                     BA, FA
"  For j = 0,...,N
"    F(FA+j)' = F(FA+j) + K*B(BA+j)
"    B(BA+j)' = B(BA+j) + K*F(FA+j)

LATRED: Y->J, X*MB:22, INC XA, INC YA   "set J=N, MA=K
        X->YA, Y->XA, DEC J             "set BA,FA
        MA*Y:FR, INC YA                 "K*B(0)
        MA*X:FR, X->T, INC XA           "K*F(0), T=F(0)
        ML+T->T, W:H->V, MA*Y:FR,       "T=F(0)', V=B(0)
                Y->W, INC YA            "K*B(1), W=B(1)
        ML+V:H->V, T:G->U,              "V=B(0)', U=F(0)'
                MA*X:FR, X->T,          "K*F(1), T=F(1)
                DEC XA, 3->YC           "set up YA INC
LTRLP:  U->X, INC XA(2), V->U,          "F(j)'=X, U=B(j)'
                ML+T->T, W:H->V,        "T=F(j+1)',V=B(j+1)
                MA*Y:FR, Y->W,          "K*B(j+2),W=B(j+2)
                DEC YA(2)
        U->Y, INC YA(YC), ML+V->V,      "B(j)'=Y
                T:F->U, MA*X:FR,        "V=B(j+1)',U+F(j+1)'
                X->T, DEC XA, DEC J,    "K*F(j+2)<T=F(j+2)
                IF J>0 GOTO LTRLP       "j+1->j
        READ, GOTO EXMAC                "exit
        END
```

```
        TITLE LDFX
        ENTRY LDFX
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

        EQU  DAX='17'O

LDFX:   W->DA(DAX)                      "set index file cell
        GOTO OPSEQ
        END
```



```
        TITLE LDSTF
        ENTRY LDF0, LDF1, LDF2, LDF3, LDF4, LDF5, LDF6,
              STF0, STF1, STF2, STF3, STF4, STF5, STF6,
              STF7, STF10, STF11, STF12, STF13, STF14,
              STF15, STF16, STF17
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO LDFj, ARG    j=0,...,6   DOP=0,2,4,6
"MACRO STFj, DA     j=0,...,17  DOP=0,2,4,6

LDF0:   W->DA(0)
        GOTO OPSEQ
LDF1:   W->DA(1)
        GOTO OPSEQ
LDF2:   W->DA(2)
        GOTO OPSEQ
LDF3:   W->DA(3)
        GOTO OPSEQ
LDF4:   W->DA(4)
        GOTO OPSEQ
LDF5:   W->DA(5)
        GOTO OPSEQ
LDF6:   W->DA(6)
        GOTO OPSEQ

STF0:   U->T, DA(0)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF1:   U->T, DA(1)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF2:   U->T, DA(2)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF3:   U->T, DA(3)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF4:   U->T, DA(4)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF5:   U->T, DA(5)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF6:   U->T, DA(6)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF7:   U->T, DA(7)->U                  "get file value
        W->DA, GOTO STF                 "set destination
STF10:  U->T, DA(10)->U                 "get file value
        W->DA, GOTO STF                 "set destination
STF11:  U->T, DA(11)->U                 "get file value
        W->DA, GOTO STF                 "set destination
STF12:  U->T, DA(12)->U                 "get file value
        W->DA, GOTO STF                 "set destination
STF13:  U->T, DA(13)->U                 "get file value
        W->DA, GOTO STF                 "set destination
STF14:  U->T, DA(14)->U                 "get file value
        W->DA, GOTO STF                 "set destination
STF15:  U->T, DA(15)->U                 "get file value
        W->DA, GOTO STF                 "set destination
```

```
STF16:  U->T, DA(16)->U                 "get file value
        W->DA, GOTO STF                 "set destination
STF17:  U->T, DA(17)->U                 "get file value
        W->DA, GOTO STF                 "set destination

STF:    XB=U, YB=U, WRITE, YB->DA,      "F->D, restore F
                T->U, GOTO OPSEQ        "restore U, exit
        END
```



```
        TITLE LOGPWR
        ENTRY LOGPWR
        EXT   FLOAT2, FLOG, OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO LOGPWR, XYA(DESC)    DESC=XYA(DATA),N
"   U = SCALE OF DATA
"   Converts 32 bit data to floating point,
"   computes the logarithm, and fixes the
"   result with a scale of 7.  The result is
"   stored in place in XY.

LOGPWR:  U->W, Y->J, X->YA, X->XC       "W=SCALE
        XC->XA, DEC J
LOOP:   Y:H->V, X->U                    "data pt in UV, SCL=W
        W->Y, U:G, V:H, CALL FLOAT2     "save SCL, float
        CALL FLOG                       "compute log
        W-'7'0->W                       "compute shift
        SHIFT(W)*V:PP                   "shift V
        MA*U:P2                         "shift U
        ML:H->V, 0->U                   "shifted V
        MLMR+UV:GH->UV, Y->W            "result in UV, get SCL
        U->X, V->Y, INC XA, INC YA,     "store result
              DEC J, IF J>0 GOTO LOOP
        GOTO OPSEQ
        END
```

```
        TITLE MOVECX
        ENTRY MOVECX
        EXT EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO MOVECX, VAL, XA, LEN   DOP=1,3,5,7

MOVECX: DY->J, DX->XC                   "set up J
        XC->XA, DEC J, W->V, V->T       "save V
        NOOP                            "wait for DEC J
        V->X, INC XA, DEC J,            "store val
                IF J>0 GOTO .
        READ, T->V, GOTO EXMAC
        END
```



```
        TITLE MOVERD
        ENTRY MOVERD
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

        EQU  DI='12'O

"MACRO MOVERD, RA, DA, N   DOP=1,3,5,7

MOVERD: DY->J                           "COUNT
        W->RA, DX->W                    "set SOURCE
        W->DA(DI:D), DEC J              "set DEST
        1->RC, U->T                     "INC=1
LOOP:   RL->U                           "get RL
        WRITE URR, INC RA, DEC J,       "store RLR
                IF J>0 GOTO LOOP
        T->U, GOTO OPSEQ                "exit
        END
```

```
        TITLE MOVES
        ENTRY MOVEDD, MOVEYD, MOVEXD,
              MOVEDX, MOVEDY, SETD,
              SMVXD, SMVYD
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO MOVEDD, DA(SOURCE), DA(DEST), COUNT  DOP=DY->W, READ
"MACRO MOVEDY, DA(SOURCE), YA(DEST), COUNT  DOP=DY->W, READ
"MACRO MOVEDX, DA(SOURCE), XA(DEST), COUNT  DOP=DY->W, READ
"MACRO MOVEXD, XA(SOURCE), DA(DEST), COUNT  DOP=DY->XA, READ
"MACRO MOVEYD, YA(SOURCE), DA(DEST), COUNT  DOP=DX->YA, READ
"MACRO SMVXD, DA, XA, CNT   DOP=1,3,5,7
"MACRO SMVYD, DA, YA, CNT   DOP=1,3,5,7
"MACRO SETD, ARG, xxx, DA   DOP=1,3,5,7

        EQU DI='12'O

MOVEDD: DY->J, V->T                     "save V
        DEC J, W->DA(DI), DX->V         "V=dest DA
DDLP:   READ, V->DA, 1+W->W,            "fetch word, DA=DEST
                1+V->V                  "inc both DA's
        DEC J, IF J>0 GOTO DDLP,        "store word
                XB=DX, WRITE,           "DA=source
                W->DA
        GOTO OPSEQ, T->V                "restore V, done

"MACRO MOVEDY, DA(SOURCE), YA(DEST), COUNT   DOP=DY->W, READ
MOVEDY: DY->J, DX->YA                   "set COUNT, YA
        DEC J, W->DA(DI)                "set DA
        READ                            "fetch first word
        DX->W, READ                     "first d->w,fetch second

        W->Y, DX->W, READ, INC YA,      "store Y(j), D(j+1)->w
                DEC J, IF J>0 GOTO .    "branch CNT-1 times
        GOTO OPSEQ                      " done

"MACRO MOVEDX, DA(SOURCE), XA(DEST), COUNT  DOP=DY->W, READ
MOVEDX: DY->J                           "set COUNT
        W->DA(DI), DEC J, DX->W         "set DA
        W->XA, READ                     "set XA, fetch word
        DX->X, READ, INC XA,            "D->X, fetch next
                DEC J, IF J>0 GOTO .    "branch CNT-1 times
        GOTO OPSEQ

"MACRO MOVEXD, XA(SOURCE), DA(DEST), COUNT   DOP=DY->XA, READ
MOVEXD: DY->J, DX->W                    "set COUNT
MVXD2:  W->DA(DI), DEC J                "set DA
        NOOP                            "wait for count
        WRITE X, INC XA, DEC J,         "X->D
                IF J>0 GOTO .           "branch CNT-1 times
        GOTO OPSEQ

"MACRO MOVEYD, YA(SOURCE), DA(DEST), COUNT   DOP=DX->DY, READ
```



```
MOVEYD: DY->J, DX->W                    "set count
MVYD1:  W->DA(DI), DEC J                "set DA
        V->T, Y->V, INC YA              "first Y
        WRITE V, Y->V, INC YA,          "V->, Y->V
                DEC J, IF J>0 GOTO .    "branch CNT-1 times
        GOTO OPSEQ, T->V                "resotre V, exit

"These macros have their parameters reversed so
"   DA can be specified indirectly.

"MACRO SMVXD, DA, XA, CNT   DOP=1,3,5,7
SMVXD:  DX->XC, DY->J                   "set J
        XC->XA, GOTO MVXD2              "set XA

"MACRO SMVYD, DA, YA, CNT   DOP=1,3,5,7
SMVYD:  DX->YA, DY->J                   "set YA,J
        GOTO MVYD1

"MACRO SETD, ARG, xxx, DA   DOP=1,3,5,7
SETD:   DY->DA(DI), U->T, V->U,         "set DEST, ARG->V
                W:H->V
        WRITE V, U:H->V, T->U,          "ARG->D
                GOTO OPSEQ              "exit
        END
```

```
        TITLE MULTL
        ENTRY MULTL
        EXT EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO MULTL,  XYA(DESC)    DESC = XYA, L

MULTL:  Y->J, X->YA, X->XC              "set up J,YA
        XC->XA, DEC J                   "set up XA
        X*Y:22, INC XA, INC YA          "start first
        NOOP
MLTLL:  X*Y:22, DEC XA, DEC YA,
                MLMR->UV
        U->X, V->Y, INC XA(2),          "store result
                INC YA(2), DEC J,       "repeat L times
                IF J>0 GOTO MLTLL
        READ, GOTO EXMAC
        END
```



```
        TITLE ORMOD
        ENTRY ORMOD
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO ORMOD,  A(DESC)     DESC = XA, L
"  Computes OR(ABS Xi)   i=0,...,L-1
"  Returns U = Number of leading zeros, V = OR

ORMOD:  Y->J, X->XC                     "set up J
        XC->XA, 0->U, 0:H->V            "first val is 0
ORLP:   ABS X->U, U OR V->V,            "Do loop L+1 times
                INC XA, DEC J,
                IF J>0 GOTO ORLP
        SCLV->U, READ, GOTO EXMAC       "leading zero cnt
        END
```

```
        TITLE POWER
        ENTRY POWER
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO POWER,  XYA(DESC)    DESC = XYA, N

POWER:  Y->J, X->XC, X->YA              "set COUNT
        XC->XA, 3->XC, DEC J            "set XA INC
        X*X, INC XA                     "X0*X0
        Y*Y, INC YA, 0->T               "Y0*Y0
        ML+T->T, MR->U, X*X, INC XA     "X1*X1
        MLMR+TU:GH->VW, Y*Y,            "X0*X0+Y0*Y0
                DEC YA, 0->T            "Y1*Y1
LOOP:   W->Y, INC YA(2), ML+T->T,       "store P(j)R
                MR->U, X*X, DEC XA(2)   "X(j+1)*x(j+1)
        V->X, INC XA(XC),               "store P(j)L
                MLMR+TU:GH->VW,         "P(J+1)
                Y*Y, DEC YA, 0->T,      "Y(j+2)*Y(j+2)
                DEC J, IF J>0 GOTO LOOP
        READ, GOTO EXMAC
        END
```



```
        TITLE RADIAN
        ENTRY RADIAN, AQUAD, BQUAD, CQUAD, DQUAD
        EXT   FMULT, FADD, FLOAT2
        EXPAND
        NOLIST
        SYMBOL

"Calculates quadrant, table fetch n, delta, sinD, cosD,
"   TRG(n), TRG(128-n), and quadrant formulas for FSIN
"   and FCOS in subroutine FCOS
"RADIAN assumes M->uv, s+2->w, v->X0, u->X1, s+2->Y0,
"   XA,YA->-, and 2/pi has been fetched

"RADIAN returns these stored XY values to the calling subroutine
"  for the determination of the proper quadrant for sin and cos,
"  with XA,YA->8, n->u, and TRG(128-n) ->tvw;  sinD is in XY1,Y2;
"  cosD is in XY3,Y4; TRG(n) is in XY5,Y6; TRG(128-n) is in XY7,Y8

"AQUAD:   cosD*TRG(n) + sinD*TRG(128-n)          XYA->6
"            sin(quad0) = cos(quad3)
"BQUAD:   cosD*TRG(128-n) + sinD*(-TRG(n))       XYA->6
"            sin(quad1) = cos(quad0)
"CQUAD:   cosD*(-TRG(n)) + sinD*(-TRG(128-n))    XYA->6
"            sin(quad2) = cos(quad1)
"DQUAD:   cosD*(-TRG(128-n) + sinD*TRG(n)        XYA->6
"            sin(quad3) = cos(quad2)

        EQU   PIDTWO='1771'O, TWODPI='1772'O, TBASE='1570'O

RADIAN: RR*V:PP, W:H, 0->T                       "begin 2/pi*M
        RL*MB:PP, INC YA                         "YA=1
        MLMR:GH->UV, RR*U:P2,
                IF H<0 GOTO RSHIFT
        MLMR+0U:GH->UV, T+GCO:F->T,
                RL*MB:P2, V->XL,
                PIDTWO->Y, DEC YA                "YA=0
        MLMR+UV:GH->UV, GASN+T+GCO:F->T,         "begin shift
                SHIFT(Y)*XL:PP, INC YA           "YA=1
        MLMR+TU:FG->TU, MA*V:PP,                 "shift v
                IF SHIFT SMALL GOTO LSM
        MA*U:PP, ML->W                           "shift u
        MA*T:P2, MLMR+0W:GH->VW,                 "shift t
                '200'O->YL                       "128=YL
        MLMR+0V:GH->UV, Y->RA, DEC YA            "fetch pi/2,YA=0
        MR+U:G->U, V*YL:PP,                      "V*128
                '777'O AND V->V                  "mask 7 lead 0
EPSL:   '400'O AND V, '400'O*'400'O:PP,
                W->YL, '400'O->XL
        U->X, ML->U, '77777'O->W                 "INTQ=X0,N=U
        IF H=0 GOTO CONT
        ML+U:G->U, V-'1000'O:H->V                "ROUND
CONT:   RR*YL:PP                                 "begin E*pi/2
        RL*YL:PP, U->Y, 0->T                     "N=Y0,E=.VW
        RR*V:P2, ML->W
        MLMR+0W:GH->VW, RL*MB:P2
        GASNML+TV+HCO:FG->TV, MR+W:H
```

```
        MLMR+TV:FG->UV, 0->W,                   "D=UV
              INC XA, INC YA                    "XYA=1
        U:G, V:H, CALL FLOAT2
        U->X, V->Y, INC XA, INC YA              "XYA=2
        W->Y                                    "D=XY12
DELTA:  XBXB-UV:GH->UV, XB=0                    "-D=UVW
        CALL FMULT                              "-D*D
        INC XA, INC YA, W-1->W                  "-D*D/2
        V->Y, U->X, INC XA, INC YA              " =XY34
        W->Y, DEC XA(2), DEC YA(2)              "XYA=2
        CALL FMULT
        XB*V:PP, XB='52525'O                    "-D*D*D/6
        MA*U:P2, '77777'O->V
        MLMR+OV:GH->UV, 0->T
        MLMR+UV:GH->UV, GASN+T+GCO:F->T
        MLMR+TU:GH->UV                          "-D*D*D/6
        CALL FADD
        W->Y, 1->W, DEC XA, DEC YA
        U->X, V->Y, 0->V,                       "SIND=XY12
              INC XA(2), INC YA(2)              "XYA=3
        '40000'O->U                             "1->UVW
        INC XA, INC YA, CALL FADD               "XYA=4
        W->Y, '11'O->T, DEC XA, DEC YA          "T=9,XYA=3
        U->X, V->Y, DEC XA(2), DEC YA(2)        "COSD=XY34
        DEC XA, DEC YA, YB->W, T->U,            "XYA=0
              YB=TBASE
        Y+W->W, Y->YL, Y->V, INC YA(2)          "YA=2
        SHIFT(SCLV)*YL:PP, X->T,                "INTQ=T
              U-SCLV->V, INC XA(2),             "NSCL=V
              INC YA(2)                         "XA=2,YA=4
        W->RA, XB-W->W, XB=TBASE+'200'O,        "fetch TRGT(N)
              INC XA(2), INC YA                 "XA=4, YA=5
        MR->U, 1+V:H->V, W->Y,                  "fix scl TRG(N)
              INC XA(2), INC YA                 "XYA=6
        RR*U:PP, T->X                           "begin TRG(N)
        RL*MB:PP, '77777'O->W
        ML+ HCO:G->V, MR+W:H, V->W              "scl TRG(N)=W
        MLMR+OV:GH->UV, CALL FLOAT2             "float TRG(N)
        X->T, W->Y, DEC XA, DEC YA              "XYA=5
        U->X, Y->W                              "128-N=W
        V->Y, XB+W:H->V, XB:G,                  "TRG(N)=XY56
              XB=TBASE, INC XA(2),              "TBASE+128-N=V
              INC YA(2)                         "XYA=7
        V->RA, W:H->V, '11'O->U,                "fetch TRGT(128-N)
              INC XA, INC YA                    "XYA=8
        SHIFT(SCLV)*V:PP, U-SCLV->W
        T->X, 1+W->W                            "fix scl TRG
        MR->U, '77777'O:H->V
        RR*U:PP                                 "begin TRG(128-N)
        RL*MB:PP
        ML+HCO:G->V, MR+V:H
        MLMR OV:GH ->UV, CALL FLOAT2
        X->T, W->Y, DEC XA, DEC YA              "INTQ=T,XYA=7
        U->X, V->Y, T:G->U, U->T,               "TRG(128-N)=XY78
              INC XA, INC YA, RTN               "INTQ=U,XYA=8
DQUAD:  OO-UV:GH->UV, DEC XA(2), DEC YA(2)              "-TRG(128-N)
```



```
        CALL FMULT                              "XYA=4
        W->Y, DEC XA, DEC YA                    "COS*TRG(128-N)
        U->X, V->Y, INC XA(2), INC YA(2)        "COS PROD=XY34
        X->U, Y:H->V, INC XA, INC YA            "XYA=6
        Y->W, DEC XA(2), DEC YA(2)              "XYA=4
        DEC XA(2), DEC YA(2), CALL FMULT        "XYA=2
        INC XA(2), INC YA(2), CALL FADD         "XYA=4
DONED:  DEC XA(2), DEC YA(2), GOTO DONEC        "XYA=2
BQUAD:  DEC XA(2), DEC YA(2), CALL FMULT        "XYA=4
        W->Y, DEC XA, DEC YA                    "COS*TRG(128-N)
        U->X, V->Y, INC XA(2), INC YA(2)        "XYA=5
        OO-XY:GH->UV, INC XA, INC YA            "-TRG(N)
        Y->W, DEC XA(2), DEC YA(2)              "XYA=4
        DEC XA(2), DEC YA(2), CALL FMULT        "XYA=2
        INC XA(2), INC YA(2), CALL FADD         "XYA=4
DONEB:  DEC XA(2), DEC YA(2), GOTO DONEC        "XYA=2
CQUAD:  OO-UV:GH->UV, DEC XA(2), DEC YA(2)      "-TRG(128-N)
        DEC XA(2), DEC YA(2), CALL FMULT        "XYA=2
        W->Y, DEC XA, DEC YA
        U->X, V->Y, INC XA(2), INC YA(2)        "SIN PROD=XY12
        INC XA(2), INC YA(2)                    "XYA=5
        OO-XY:GH->UV, INC XA, INC YA            "-TRG(N)
        Y->W, DEC XA(2), DEC YA(2),             "XYA=4
              CALL FMULT
        DEC XA(2), DEC YA(2), CALL FADD         "XYA=2
DONEC:  DEC XA(2), DEC YA(2), RTN               "XYA=0
AQUAD:  DEC XA(2), DEC YA(2)                    "XYA=4
        DEC XA(2), DEC YA(2), CALL FMULT        "XYA=2
        W->Y, DEC XA, DEC YA
        U->X, V->Y, INC XA(2), INC YA(2)        "SIN PROD=XY12
        X->U, Y:H->V, INC XA, INC YA            "XYA=4
        Y->W, INC XA(2), INC YA(2),             "XYA=6
              CALL FMULT
        DEC XA(2), DEC YA(2)                    "XYA=4
        DEC XA(2), DEC YA(2), CALL FADD         "XYA=2
DONEA:  DEC XA(2), DEC YA(2), RTN

RSHIFT: MLMR+OU:GH->UV, T+GCO:F->T,
              RL*MB:P2, PIDTWO->Y, DEC YA       "YA=0
        MLMR+UV:GH->UV, '200'O->YL,
              GASN+T+GCO:F->T                   "128=YL
        MLMR+TU:FG->TU, 5->XL,
              5+W:H
        SHIFT(Y)*U:PP, INC YA(2)                "shift u,YA=2
        MA*T:P2, DEC YA,                        "shift t, YA=1
              IF H<0 GOTO DELFLT
        ML->W, Y->RA, DEC YA, T:G->U            "fetch pi/2
        MLMR+OW:GH->VW, '777'O->T
        T AND V->V, 0->U, V*YL:PP,              "MASK 7 LEAD 0
              IF G<0 GOTO RQ3
        GOTO EPSL
RQ3:    3->U
        GOTO EPSL
DELFLT: X->V, 0->T, INC XA                      "XA=1
        X->U, W-2->W, DEC XA, DEC YA            "S=W, XYA=0
        T->X, 'O'O->Y                           "O=N=INTQ
```

```
        IF G>0 GOTO Q0
Q3:     3->X                                "INTQ=3=X0
        '200'0->Y                           "N=128=Y0
Q0:     INC XA, INC YA, U:G, V:H,
                CALL FLOAT2
        U->X, V->Y, INC XA, INC YA          "XYA=2
        W->Y, GOTO DELTA

LSM:    MA*U:PP                             "shift u
        MA*T:P2, ML->W, '200'0->YL          "shift t,128=YL
        MLMR+0W:GH->VW, '777'0->T,          "YA=0
                Y->RA, DEC YA               "fetch pi/2
        MLMR+0V:GH->UV
        T AND V->V, V*YL:PP, GOTO EPSL
        END
```



```
        TITLE RANDOMS
        ENTRY RANDOMS
        EXPAND
        NOLIST
        SYMBOL

"MACRO RANDOMS,  XYA(DESC)   DESC = R(-1), RM
"                                   SCL,  N
"                                       , XA
"  Computes Rj = R(j-1)*RM  (Modulo 2**16)
"    and X(XA+j) = Rj*SCL:FR   for j=0,...,N

RANDOMS:  X*Y:PP, Y->YL, XA->XC,        "start R(-1)*RM
                INC XA, INC YA          "YL=RM
        X->XL, Y->J, INC YA             "set J, XL=SCL
        MR->V, DEC J, Y->XA             "V=R(0)
        V*YL:PP                         "start R(0)*RM
        V*XL:FR                         "start R(0)*SCL
RLP:    MR->V, V->T                     "T=R(j), V=R(j+1)
        V*YL:PP, ML->U                  "U=X(j)
        U->X, INC XA, V*XL:FR,          "store X(j)
                DEC J, IF J>0 GOTO RLP  "j+1->j, do N times
        XC->XA, READ                    "pt at R(-1) save
        T->X, EXEC MACRO                "save R(N-1)
        END
```

```
        TITLE RECORD
        ENTRY RECORD
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO RECORD  XYA(CHNLBLK)  CHNLBLK = AREC, MREC
"                                  BLNK, PA
"                                  PAV , PB
"                                  DMP , PC

        EQU  EXBS='1420'O,          "16 word table in Y
             EXPEND='1440'O,        "limit
             PMAX='127'D,           "Max PAV value
             PMIN='36'D,            "Min PAV value
             DF='12525'O,           "Damping table mult
             BF='31460'O,           "Blanking multiplier
             DTABS='1420'O          "8 word table in X

RECORD: U-X->U, INC XA,             "ACUR-AREC
                EXBS*MB:PP          "base of EXP DECAY TABLE
        U-X->U, U->W, INC XA(2)     "U=ADISP, W=new PA
                MA*1:PP             "EXPBS*1
        U*X:FR, DEC XA              "start DMP*ADISP
        IF G>0 GOTO NOTBLNK,        "ADISP>0?
                MR->U               "U=EXBS
EXIT:   READ, GOTO EXMAC            "no, exit
NOTBLNK:  ML*U->U, EXPEND->T        "U=EXPXA
        U->XA, XA->XC, U-T:G,       "fetch EXPT value
                '100000'O->XL       "EXPXA-EXPEND
        X*Y:FR, XC->XA, V->U        "MREC*EXPT,U=MCUR
        XL*XL:P2, X+W:H->V,         "MA=0.5,V=PAV+PA
                IF G<0 GOTO CHKTH   "EXPXA<EXPEND?
        NOOP
CHKTH:  U-ML:G, PMAX:H->V,          "MCUR-THRESH
                MA*V:PP             "start 0.5*(PAV+PA)
        PMIN->T
        IF G<0 GOTO EXIT, V-ML:G    "PMAX-PAV
        U->Y, INC YA, ML-T:G,       "MCUR-.MREC,PAV-PMIN
                ML:H->V, DF*MB:PP   "V=PAV
        MA*V:FR, V->X, Y->U,        "U=old PA
                DEC XA(2),          "PAV*DF
                IF G<0 GOTO UPMAX   "PAV>PMAX?
        W->Y, INC YA, X+W:H->V,     "new PA->Y, V=ACUR
                INC XA(2),          "PAV>PMIN?
                IF G>0 GOTO CONT
        MA*PMIN:FR, PMIN->X         "no, PMIN->PAV
CONT:   Y->W, XB=DTABS-'6'O,        "W=old PB
                XB->T, DEC XA(2)
        U->Y, INC YA, ML+T->T,      "PA'->PB, T=YA(DMP)
                BF*MB:FR            "compute BLNK
        W->Y, T->YA                 "PB'->PC, fetch DMP
        V->X, ML->U, Y:H->V, INC XA "ACUR->AREC, V=DMP
        U->X, INC XA(2), READ       " new BLNK
        V->X, EXEC MACRO            "new DMP
```



```
UPMAX:  X->T, INC XA(2), PMAX->V    "T=AREC, pt at PAV
        GOTO CONT, MA*V:FR, V->X,   "DF*PMAX, PMAX->PAV
                W->Y, W+T->V, INC YA "PA->Y, V=ACUR
        END
```

```
        TITLE RMOVES
        ENTRY RMVXY, RMVYX
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO RMVXY, XA, YA, N   X(XA+j)->Y(YA-j);N
"MACRO RMVYX, YA, XA, N   Y(YA+j)->X(XA-j);N

RMVXY:  DX->YA, DY->J                   "set YA,J
        DEC J, X->W, INC XA             "W=X(XA+0)
        NOOP
        W->Y, DEC YA, X->W, INC XA,     "X(+j)->Y(-j)
              DEC J, IF J>0 GOTO .      "W=X(+j+1), j+1->j
        READ, GOTO EXMAC                "exit

RMVYX:  DX->XC, DY->J                   "set J=N

        DEC J, XC->XA, V->W             "set XA, save V
        Y->V, INC YA                    "V=Y(+0)
        V->X, DEC XA, Y->V, INC YA,     "store X(-j)
              DEC J, IF J>0 GOTO .      "V=Y(+j+1), j+1->j
        W->V, READ, GOTO EXMAC          "exit
        END
```



```
        TITLE SAVE
        ENTRY SAVE, RESTORE
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO SAVE, DA   DOP=ARG->DA, DBLE  U,V,XA,YA->D
"MACRO RESTORE, DA  DOP=ARG->DA, DBLE  D->U,V,XA,YA

        EQU  EA='31'O

SAVE:   XA->XC, YA->YC, WRITE UV
        WRITE XCYC, GOTO OPSEQ
RESTORE:  READ
        DX->U, DY->V, READ
        DX->XC, DY->YC, READ (EA)
        XC->XA, YC->YA, EXEC MACRO
        END
```

```
        TITLE SCHED
        ENTRY SCHED
        EXT   OPSEQ
        EXPAND
        NOLIST
        SYMBOL

"MACRO SCHED, A(ROUTINE)    DOP=0,2,4,6
"  Add ROUTINE to subroutine stack at bottom +1
"  Must not be used in to level routine

        EQU  BOS='12'O, STK='10'O

SCHED:  V->T, W:H->V, DA(STK)->W        "get current ptr
        YB=BOS+1, YB->DA, W-YB->W               "W=CNT-1
        W->J, 0->W                      "set counter
        READ, INC DA(W), GOTO SLPE      "fetch entry
SLP:    DX->V, READ, INC DA(0)          "V=prev stk entry
SLPE:   WRITE V, DEC J,                 " move entry up
                IF J>0 GOTO SLP
        T->V, GOTO OPSEQ
        END
```



```
        TITLE SCORE
        ENTRY SCORE
        EXT EXMAC
        EXPAND
        NOLIST
        SYMBOL

"  This program computes a "best choice" pitch value
"  and its "score" using the Gold-Rabiner algorthm.
"  It uses an input table of 36 pitch values
"  prepared by CHANLIZE from the list of extremes
"  in the low-passed speech data, along with the
"  pitch selected for the previous frame.  Each of 6
"  pitch candidates is checked using four successively
"  larger windows.

"MACRO SCORE,  xxx
"  This program is entered at SCORE, below

        EQU   BIPTR='174'O, POPTAB=BIPTR+4,
              CNDEND=POPTAB+'36'D, CAND=CNDEND+1,
              CNDPTR=CAND+1, BTAB=CNDPTR+1, CTAB=BTAB+2

TOP:    V->Y, U-X:G, INC XA, DEC YA     "CNDPTR->Y,CAND-CTAB(1)
LOOP:   U->Y, XA->XC, U-X:G, INC XA,    "U-X(j),j->XC,CAND->Y
                IF G>0 GOTO LOOP        "repeat if CAND>CTAB(j-2)
        XC->YA, 0->V, BIPTR->XA         "fetch DPN, pt at BIPTR save
        BTAB->X, BTAB->T, INC XA(2)     "init. BIPTR
        Y->U, V->X, INC XA              "PANE=0, U=DPN
        3->J, T->YA                     "Y=BIAS, set J
        U->X, DEC XA, IF G=0 GOTO SCORE "DPN->X, CAND no good?

"  The following loop is repeated four times with
"  windows = DPN, 2*DPN, 3*DPN, and 4*DPN.
"  Assume  X = PANE, Y = BIAS

NXTPN:  X->V, INC XA, CAND->YC          "V=old PANE
        X+V->V, DEC XA, Y->W, YC->YA    "V=new PANE, W=BIAS
        V->X, Y+V->U, Y-V:H->V,         "U=UB, V=LB
                DEC YA(2)
        V*Y, INC XA(2), W:H->V          "LB->MR, V=init count
SETUP:  X->T, U-X:H, INC XA             "T=Pj, UB-Pj
        X->T, U-X:H, INC XA,            "H=UB-Pj, Pj->t
                MR-T:G, GOTO CNTLP      "G=LB-P(j-1)
RSET:   X->T, U-X:H, INC XA,            "H=UB-Pj, Pj->T
                MR-T:G,                 "G=LB-P(j-1)
                IF G<0 GOTO FIN         "done?
CNTLP:  X->T, U-X:H, INC XA,            "U=UB-Pj, Pj->T
                MR-T:G,                 "P(j-2)>UB?
                IF H<0 GOTO CNTLP
        X:G, DEC XA(2), Y+V:H->V,       "may be in, inc cnt
                IF G<0 GOTO SETUP       "if P(j-2)>LB count it
        X->T, U-X:H, INC XA,            "restart UB-Pj-1
                V-Y->V, GOTO RSET       "dec count
FIN:    BIPTR->XC, Y->U, V:H->W,        "U=1, W=count
                INC YA
```

```
        Y-W:H, INC YA, XC->XA           "score count
        X+U->U, Y:H->V, DEC YA          "V=CAND
        U->X, INC XA,                   "update bias ptr
                IF H>0 GOTO NOTBST
        V->X, W->Y                      "CAND->WINNER,COUNT->SCORE
NOTBST:  INC XA, U->YA, DEC J,          "Y=BIAS, X=PANE
                IF J>0 GOTO NXTPN       "done?

"  ENTRY POINT FOR SCORE and for each candidate PITCH

SCORE:  CNDEND->W                       "limit value
        CNDPTR->YA                      "fetch XA(CAND)
        Y->XA, Y-W:H, '6'O:G->V         "CNDPTR-CNDEND
        X->U, CTAB->XA                  "U=new CAND
        Y+V:H->V, U-X:G, INC XA,        "V=new CNDPTR,CAND-CTAB(O)
                IF H<O GOTO TOP         "have a candidate
        READ, GOTO EXMAC
        END
```



```
        TITLE SFLT
        ENTRY SDIV, SFLT
        EXT   FDIVS, FLOAT2, EXMAC
        EXPAND
        NOLIST
        SYMBOL

"SHORT FLOATING POINT OPERATIONS.
"  All operations assume the first operand is in U/V,
"  U=Mantissa, V=Scale.
"  The second operand is in XY at the location given
"  by the argument.  X=Mantissa, Y=Scale

"MACRO SFLT, SCL.  Converts UV*2**SCL to short FP.

SFLT:   U:G, V:H, CALL FLOAT2           "test value, float
        W->V, READ, GOTO EXMAC          "V=SCL, exit

"SDIV, ARG  DOP=0,2,4   OP2 at XY(ARG)
"  Computes X/Y  /  U/V -> U/V

SDIV:   T->YA, W->XA, V:H->W,           "XYA(OP2)
                CALL FDIVS
        W:H->V, READ, GOTO EXMAC
        END
```

```
        TITLE SHORTOPS
        ENTRY LDUX, LDVY, LDUI, LDUY, LDU, LDV, LDUVXY, LDUVD,
              STUX, STVY, STVX, STUY, STUVXY, STUXI, STUYI,
              STUD, STVD, STUVD, LDX, LDY, LDIY, LDXY, LDUYV,
              ADDU, ADDUX, SUBU, SUBAU, SUBV, SUBAV, ADDV,
              ADDVY, ADDAY, ADDAX, MULTU, MULTV, FMULTU, FMULTV,
              IFUEQ, IFUGT, IFULT, IFVLT, TLYY, IFYLT, IFALT

        EXT   EXMAC, OPSEQ, GOTO
        EXPAND
        NOLIST
        SYMBOL

"A collection of short operations involving UV and an argument

        EQU   EA='31'O

"MACRO LDUX, XA  DOP=13  X(XA)->U
LDUX:   X->U, EXEC MACRO

"MACRO LDVY, YA  DOP=15  Y(YA)->V
LDVY:   Y->V, EXEC MACRO

"MACRO LDUI, YA  DOP=14  X(Y(YA))->U
LDUI:   Y->XA, READ, GOTO LDUX

"MACRO LDUY, YA  DOP=15   Y(YA)->U
LDUY:   Y->U, EXEC MACRO

"MACRO LDU,  ARG  DOP=1,3,5,7  ARG->U
LDU:    W->U, EXEC MACRO

"MACRO LDV,  ARG   DOP=1,3,5,7  ARG->V
LDV:    W->V, EXEC MACRO

"MACRO LDUVXY,  XYA  DOP=11  XY(XYA)->UV
LDUVXY:  X->U, Y:H->V, EXEC MACRO

"MACRO LDUVD,  DA  DOP=0  (DA must be even)  D->UV
"    (also could have DOP=3 for inline ARG)
LDUVD:  DX->U, DY:H->V, READ, GOTO EXMAC

"MACRO STUX, XA  DOP=13   U->X(XA)
STUX:   U->X, EXEC MACRO

"MACRO STVY, YA  DOP=15  V->Y(YA)
STVY:   V->Y, EXEC MACRO

"MACRO STVX,  XA  DOP=13  V->X(XA)
STVX:   V->X, EXEC MACRO

"MACRO STUY,  YA  DOP=15  U->Y(YA)
STUY:   U->Y, EXEC MACRO

"MACRO STUVXY, XYA  DOP=11  UV->XY(XYA)
STUVXY: U->X, V->Y, EXEC MACRO
```



```
"MACRO STUXI, YA  DOP=14  U->X(Y(YA))
STUXI:  Y->XA, READ, GOTO STUX

"MACRO STUYI, YA  DOP=14  U->Y(Y(YA))
STUYI:  Y->YC
        YC->YA, READ, GOTO STUY

"MACRO STUD,  DA  DOP=22  U->D(DA)
STUD:   WRITE U, GOTO OPSEQ

"MACRO STVD,  DA  DOP=22   V->D(DA)
STVD:   WRITE V, GOTO OPSEQ

"MACRO STUVD,  DA   DOP=23  UV->D(DA)  DA must be even
STUVD:  WRITE UV, GOTO OPSEQ

"MACRO LDX,  ARG, XXX, XA  DOP=1,3,5,7  ARG->X(XA)
LDX:    DY->XA, READ, U->T, V->U, W:H->V
        V->X, U:H->V, T:G->U, EXEC MACRO        "ARG->X

"MACRO LDY, ARG, YA, XXX  DOP=1,3,5,7   ARG->Y(YA)
"MACRO LDIY, ARG    DOP=1,3,5,7   ARG->Y
LDY:    DX->YA, READ
LDIY:   W->Y, EXEC MACRO

"MACRO LDXY, XYA, XVAL, YVAL  DOP=11
"     XVAL->X(XYA), YVAL->Y(XYA)
LDXY:   DX->X, DY->Y, READ, GOTO EXMAC

"MACRO LDUYV, ARG  DOP=0,2,4,6  Y(ARG+V)->U
LDUYV:  V+W:H->V, V->W                  "base + index
        V->YA, W->V, READ, GOTO LDUY    "point at Y value

"MACRO ADDU, ARG  DOP=1,3,5,7   U+ARG->U
ADDU:   W+U->U, EXEC MACRO

"MACRO ADDUX,  XA  DOP=13    X+U->U
ADDUX:  X+U->U, EXEC MACRO

"MACRO ADDV,  ARG   DOP=1,3,5,7  V+ARG->V
ADDV:   W+V->V, EXEC MACRO

"MACRO ADDVY,  YA  DOP=15  Y(YA)+V->V
ADDVY:  Y+V->V, EXEC MACRO

"MACRO SUBU,  ARG  DOP=1,3,5,7  U-ARG->U
SUBU:   U-W->U, EXEC MACRO

"MACRO SUBAU,  ARG DOP=1,3,5,7  ARG-U->U
SUBAU:  W-U->U, EXEC MACRO

"MACRO SUBV,  ARG   DOP=1,3,5,7  V-ARG->V
SUBV:   V-W->V, EXEC MACRO

"MACRO SUBAV,  ARG   DOP=1,3,5,7  ARG-V->V
```

```
SUBAV:  W-V->V, EXEC MACRO

"MACRO ADDAY,  ARG, YA, XXX   DOP=1,3,5,7
ADDAY:  DX->YA
        Y+W->W, READ, GOTO LDIY

"MACRO ADDAX, VAL, XXX, XA   DOP=1,3,5,7  X(XA)+ARG->X
ADDAX:  DY->XA
        X+W:H->V, V->W, READ
        V->X, W->V, EXEC MACRO

"MACRO MULTU,  ARG   DOP=0,2,4,6   U*ARG->UV
MULTU:  U*W:22, GOTO MPUV

"MACRO MULTV,  ARG   DOP=0,2,4,6   V*ARG->UV
MULTV:  V*W:22
MPUV:   READ(EA)
        MLMR->UV, EXEC MACRO

"MACRO FMULTU,  ARG   DOP=0,2,4,6   U*ARG:FR->U
FMULTU: U*W:FR
MPU:    READ(EA)
        ML->U, EXEC MACRO

"MACRO FMULTV, ARG   DOP=0,2,4,6   V*ARG:FR->V
FMULTV: V*W:FR
MPV:    READ(EA)
        ML->V, EXEC MACRO

"MACRO TLYY,  ARG, YA,  BRANCH   DOP=1,3,5,7
"  Y(YA)+ARG->Y,  IF Y\=0 GOTO BRANCH
TLYY:   DX->YA
        Y+W->W
        W->Y
        DY->W, READ, IF H=0 GOTO EXMAC
        W->DA, GOTO OPSEQ               "set new EA

"MACRO IFYLT,  ARG, YA, BRANCH   DOP=1,3,5,7
"   IF Y(YA)<ARG,  GOTO BRANCH
IFYLT:  DX->YA
        Y-W:H
LTT:    NOOP                            "wait for test value
        DY->W, READ, IF H<0 GOTO GOTO
        EXEC MACRO

"MACRO IFALT,  ARG, VAL, BRANCH   DOP=1,3,5,7
"   IF ARG<VAL, GOTO BRANCH
IFALT:  W-DX:H, GOTO LTT

"MACRO IFUEQ, VAL   DOP=0,2,4,6
"   IF U=ARG, DO NEXT MACRO
IFUEQ:  U-W:H
        NOOP
        READ, IF H=0 GOTO EXMAC
        READ, GOTO EXMAC                "skip one doubleword
```



```
"MACRO IFULT, VAL   DOP=0,2,4,6
"   IF U<ARG  DO NEXT MACRO
IFULT:  U-W:H
ULT:    NOOP
        READ, IF H<0 GOTO EXMAC
        READ, GOTO EXMAC

"MACRO IFUGT,  ARG   DOP=0,2,4,6
"   IF U>ARG  DO NEXT MACRO
IFUGT:  W-U:H, GOTO ULT

"MACRO IFVLT, ARG   DOP=0,2,4,6
"   IF V<ARG  DO NEXT MACRO
IFVLT:  V-W:H, GOTO ULT
        END
```

```
        TITLE SMVXYD
        ENTRY SMVXYD
        INTRLV
        EXT MVXYD2

SMVXYD  DX->XC, DX->YA, DY->J          "SET YA AND COUNT
        XC->XA, GOTO MVXYD2            "SET XA, GOTO MOVE
        END
```



```
        TITLE STEPN
        ENTRY STEPN, IFEA, IFA, INCD, ONESTEP
        EXT   OPSEQ, EXEC, GOTO
        EXPAND
        NOLIST
        SYMBOL

"CMD MACRO IFA, ARG, VAL, N    DOP=1,3,5,7
"CMD MACRO IFEA, EAVAL, xxx, N  DOP=1,3,5,7
"CMD MACRO STEPN, N    DOP=DY->W
"    MACRO INCD, ARG, DA, EAVAL    DOP=1,3,5
"     D(DA)+ARG->D, IF RESULT \=0 GOTO EAVAL

        EQU  EA='31'O, DI='12'O

IFA:    DX-W:H, GOTO COMP              "VAL-ARG
IFEA:   DA(EA)-W:H                     "EA-EAVAL
COMP:   YB=DA(DI)                      "EA=DA
        IF H=0 GOTO EXEC, DY->W        "CNT->W, DONE?
STEPN:  U->T, V->U, -1+W:H->V,         "save U,V
                -1->W                  "DEC CNT
        READ (DI), INC DA(W)           "DEC FILE(DA)
        IF H<0 GOTO EXEC, WRITE V,     "update CNT, done?
              U->V, T:F->U             "restore U,V
        READ (EA)
ONESTEP:  EXEC MACRO, ENABLE           "do one MACRO

"MACRO INCD, ARG, DA, EAVAL    DOP=1,3,5
"  D(DA)+ARG->D, If result \=0 goto EAVAL

INCD:   DX->W                          "W=DA
        W->DA(DI), T->U, U->T          "save U, U=ARG
        READ, DY->W                    "fetch D, W=EAVAL
        DX+U->U, READ, INC DA(-1)      "D+ARG, back up DA
        WRITE U                        "update D
        IF G=0 GOTO OPSEQ, T->U        "is result=0
        GOTO GOTO                      "no, branch
        END
```

```
        TITLE STKOPS
        ENTRY LDSTKD, STSTKD, STKADD, POPSTK,
              STKMULT, SAVSTK, UVTOSTK, STKDIV, STKLOG
        EXT   FADD, OPSEQ, EXMAC, FLOAT, FMULT, FINV, FSUB, FLOG
        EXPAND
        NOLIST
        SYMBOL

        EQU  DI='12'O, EA='31'O

"FLOATING POINT STACK FORMAT
"            X ,  Y
"  BOS:     --  ,  SCL
"            ML ,  MR  for first number
"           --  ,  SCL2
"            ML ,  MR  for second number
"           --  ,  SCL  for TOS number (XYA points here)
"      U=ML ,  V=MR  for TOS number

"MACRO LDSTKD, A(FP WORD)  DOP=DY->F(DA)
LDSTKD: DEC XA, DEC YA, READ              "fetch ML
        DX->T, READ, V->Y, DEC YA         "ML->T, OLD MR->Y
        DX->V, READ                       "MR->V, fetch SCL
        DX->W, READ (EA)                  "SCL->W
        U->X, DEC XA, T->U, W->Y,         "old ML->X, ML->U
              EXEC MACRO                  "SCL->Y, do next

"MACRO STSTKD, DA   DOP=DY->F(DA)
STSTKD: WRITE U, Y->U                     "store ML
        WRITE V                           "store MR
        WRITE U                           "store SCL
POPSTK: INC XA, INC YA, READ (EA)         "back up prts
        X->U, Y:H->V, INC XA,             "get new TOS Man
              INC YA, EXEC MACRO

"MACRO SAVSTK, DA  push STK and save ptr  DOP=DY->F(DA)
SAVSTK:  DEC XA, DEC YA, Y->W             "W=SCL
        U->X, V->Y, DEC XA, DEC YA        "push mantissa
        XA->XC, W->Y                      "STK PTR->XC
        WRITE XC, GOTO OPSEQ              "save in D

"MACRO UVTOSTK, A,A   DOP=0,2,4   FLOAT UV->TOS
UVTOSTK:  T->YA, W->XA,                   "get stack address
              CALL FLOAT
        W->Y, READ, GOTO EXMAC            "save SCL

"MACRO STKADD , xxx   DOP=NOOP
STKADD:  Y->W, INC XA(2), INC YA(2),
              CALL FADD                   "do add
        W->Y, READ, GOTO EXMAC            "save new SCL

"MACRO STKSUB , xxx   DOP=NOOP
STKSUB:  Y->W, INC XA(2), INC YA(2),
              CALL FSUB                   "do subtract
        W->Y, READ, GOTO EXMAC            "save new SCL
```



```
"MACRO STKMULT, xxx   DOOP=NOOP
STKMULT:  Y->W, INC XA(2), INC YA(2),
              CALL FMULT                  "point at ARG2
        W->Y, READ, GOTO EXMAC            "save new SCL

"MACRO STKDIV,  xxx   DOP=NOOP
STKDIV:  Y->W, INC XA(2), INC YA(2),      "point at numerator
              CALL FINV                   "do divide
        CALL FMULT
        W->Y, READ, GOTO EXMAC            "save new SCL

STKLOG:  Y->W, CALL FLOG
        W->Y, READ, GOTO EXMAC
        END
```

```
        TITLE TSTTOL
        ENTRY TSTTOL
        EXT EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO TSTTOL, ARG
"  IF 1ARG - U1 < V,  DO NEXT MACRO, ELSE SKIP

TSTTOL: U-W->U, U->T                    "U-ARG->U
        V-U:G, V+U:H, T:F->U            "V-U, V+U in GH
        IF G<0 GOTO HTEST
        IF G>0 GOTO EXMAC, READ         "do next macro
        READ, GOTO EXMAC                "else skip
HTEST:  IF H>0 GOTO EXMAC, READ
        READ, GOTO EXMAC
        END
```



```
        TITLE UPCHAN
        ENTRY UPCHAN
        EXT   EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO UPCHAN  XYA(CHNLBLK)  CHNLBLK = AREC, A(UPDESC)
"       ACUR in U                     MREC, PA
"       MCUR in V                         , PB
"                                         , PC
"UPDATE DESC in XY = BLNK, A(EXPTAB)
"                    DMP, PAV
"                       , PS

        EQU   EXPLEN='17'O

UPCHAN: U-X->U, XA->XC, Y->XA,          "ACUR-AREC
                Y->YC                   "XA=YC=A(UPDESC)
        U-X->U, U->W, INC XA,           "U=A disp, W=PA'
                YC->YA, YA->YC          "Y=A(EXTAB)
        X*U:FR, XC->XA                  "start DMP*ADISP
        IF G>0 GOTO NOTBLNK,            "ADISP>0?
                Y->U, YC->YA, INC XA    "U=EXBS
EXIT:   READ, GOTO EXMAC
NOTBLNK: ML+U->U, EXPLEN->T             "U=EXPXA
        MA*X:FR, U->XA, T-ML:G          "MB=MREC, EXPLEN-DMP*ADISP
        X*MB:FR, XC->XA, V->U,          "MREC*EXPT, U=MCUR
                INC YA
        Y->V, INC XA,                   "V=PA
                IF G<0 GOTO UPDATE      "ADISP=DMP>EXPLEN?
        U-ML:G                          "MCUR-THRESH
        NOOP
        IF G<0 GOTO EXIT                "MCUR too small
UPDATE: U->X, DEC XA, V->U,             "MCUR->MREC
                W->Y, INC YA            "U=PA, PA'->PA
        X+W:H->V, Y->W                  "V=ACUR, W=PB
        V->X, U->Y, INC YA, READ        "ACUR->AREC,PA->PB
        W->Y, EXEC MACRO                "PB->PC, done
        END
```

```
        TITLE XYMADD
        ENTRY XMADDY, YMADDX
        EXT EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO XMADDY, XYA(DESC)  DOP=10  DESC = M, L
"                                       YA, XA
"MACRO YMADDX, XYA(DESC)  DOP=10  DESC = M, L
"                                       YA, XA

XMADDY: Y->J, X*MB:22,                  "set up U,M
              INC XA, INC YA
        Y->XA, X->YA, DEC J             "point at data
        MA*X:FR, INC XA                 "start first
        Y->W, INC YA                    "fetch old Y
XMAYL:  ML+W:H->V, MA*X:FR,             "V=result
              Y->W, DEC YA, INC XA      "start next, next old Y
        V->Y, INC YA(2), DEC J,         "store result
              IF J>0 GOTO XMAYL         "repeat L times
        READ, GOTO EXMAC
YMADDX: Y->J, X*MB:22, INC XA, INC YA   "set up U,M
        Y->XA, X->YA, DEC J             "point at data
        MA*Y:FR, INC YA                 "start first
        X->T, INC XA                    "fetch old X
YMAXL:  ML+T->U, MA*Y:FR, X->T,         "U=resutl, start next
              DEC XA, INC YA            "next old X
        U->X, INC XA(2), DEC J,         "store result
              IF J>0 GOTO YMAXL         "repeat L times
        READ, GOTO EXMAC
        END
```



```
        TITLE XYMOVES
        ENTRY MOVEXYD, MOVEDXY, MOVEXY, MOVEYX, MVXYD2
        EXT   OPSEQ, EXMAC
        EXPAND
        NOLIST
        SYMBOL

"MACRO MOVEDXY, DA(SOURCE), XYA(DEST), COUNT   DOP=DY->W, READ
"MACRO MOVEXYD, XYA(SOURCE), DA(DEST), COUNT   DOP=DY->XA,DX->YA,READ
"MACRO MOVEXY, XA, YA, COUNT   DOP=DY->XA, READ
"MACRO MOVEYX, YA, XA, COUNT   DOP=DX->YA, READ

        EQU   DI='12'O

MOVEDXY: DX->YA, DY->J                  "set COUNT
        W->DA(DI:D), DEC J, DX->W       "set DA
        W->XA, READ                     "set XA, fetch word
        DX->X, DY->Y, INC XA, INC YA,   "D->XY
               READ, DEC J,             "branch CNT-1 times
               IF J>0 GOTO .
        GOTO OPSEQ                      "EXIT

"MACRO MOVEXYD, XYA(SOURCE), DA(DEST), COUNT  DOP=DY->XA,DX->YA,READ
MOVEXYD:  DY->J, DX->W                  "set CNT
MVXYD2: W->DA(DI:D), DEC J              "set DA
        NOOP                            "wait for count
        WRITE XY, INC XA, INC YA,       "next pair
              DEC J, IF J>0 GOTO .      "branch CNT-1 times
        GOTO OPSEQ

"MACRO MOVEXY, XA, YA, COUNT  DOP=DY->XA, READ
MOVEXY: DY->J, DX->YA                   "set YA, COUNT
        DEC J
        X->W, INC XA                    "get first word
        W->Y, INC YA, X->W, INC XA,     "store word, get next
              DEC J, IF J>0 GOTO .      "branch CNT-1 times
        READ, GOTO EXMAC

"MACRO MOVEYX, YA, XA, COUNT   DOP=DX->YA, READ
MOVEYX:  DY->J, DX->W                   "set count
        DEC J, W->XA
        V->W, Y->V, INC YA              "save, first word
        V->X, INC XA, Y->V, INC YA,     "store word, get next
              DEC J, IF J>0 GOTO .      "branch CNT-1 times
        W->V, READ, GOTO EXMAC          "restore V
        END
```

APPENDIX B

```
"ASLDATA - INPUT AND OUTPUT IN BASE 64
        TITLE   ASLDATA
        LISTOBJ
        LISTXT
        ENTRY   ASLDRSP,RLSEIAL,RCVINT,OUTQASL,OUTASL

        DECLARE X: ldesca='1760'O, lflag='1763'O, outptr='1761'O
        DECLARE XY: ldesc='1761'O,stringd='1762'O,xywk='1760'O,
                        rexits='1764'O, accum='1762'O
        DECLARE Y: rcvxt='1764'O,lcode='1761'O,trxt='1763'O,
                        shifter='1761'O
        INSERT  DMEMEQ2.ins
        INSERT  EQANAD.ins
ASLDRSP SAVE    INTSVE
        MOVEXYD xywk, XYSAVE, 5          "SAVE XYMEM USED
aslrsp2 ASINT   LINEO, rcvint, trint     "DETERMINE TYPE
        MOVEDXY XYSAVE, xywk, 5          "RESTORE XYMEM
        RESTORE INTSVE                   "AND REGS
        INTRTN                           "EXIT
rcvint  STVX    ldesca                   "SAVE A(LINE DESC)
        MOVEDXY ldesca, ldesc, 4         "GET LINE DESC
        IFUEQ   '23'O
        GOTO    noint                    "XOFF?
        IFUEQ   '21'O
        GOTO    intok                    "XON?
        IFULT   '40'O
        GOTO    aslrsp2                  "IGNORE OTHER CTRL
        CALL    rcvxt                   "CALL ROUTINE
        GOTO    aslrsp2                  "GO CHECK FOR MORE
noint   ADDAX   -2, lflag                "CHANGE ACTIVITY FLAG
        GOTO    store                    "UPDATE LINE DESC
intok   LDUX    lflag                      "GET ACTIVITY FLAG
        IFUEQ   0
        GOTO    aslrsp2                  "IGNORE XON IF IDLE
        ADDU    2                         "UPDATE FLAG
        STUX    lflag
        IFULT   1
        GOTO    store                    "NOT READY?
        LDX     1, lflag                 "SET TO STATE ONE
        LDU     '47'O                         "ENABLE TXRDY
        SNDCMD  lcode                   "SET UART CTRL
        GOTO    store                      "UPDATE LINE DESC
trint   STVX    ldesca                        "SAVE A(LINE DESC)
        MOVEDXY ldesca, ldesc, 4        "GET DESC.
        IFALT   lflag, 1, trdn          "No activity
        LDUDA   X[stringd]
        ADDAX   1, X[stringd]              "UPDATE POINTER
        SNDCHR  lcode                   "SEND CHAR IN U
        TLYY    -1, Y[stringd], store          "DEC COUNT
        CALL    trxt                    "COMPLETION ROUTINE
store   SMVXYD  ldesca, ldesc, 4         "UPDATE DESC
        GOTO    aslrsp2                    "GO CHECK FOR MORE
trdn    LDU     '46'O
        SNDCMD  lcode                   "DISABEL TRDY INT
```

```
        GOTO    aslrsp2
"TOP LEVEL ASL INPUT
RCVINT  IFUEQ   playn
        GOTO    inpars          "Parcel?
        IFUEQ   addr
        GOTO    saddr             "SET ADDRESS?
        IFUEQ   data
        GOTO    data1             "DATA BLOCK?
        IFUEQ   freqn
        GOTO    freqn1          "ANAL START?
        IFUEQ   stop
        GOTO    stop1             "Anal stop?
        IFUEQ   outd
        GOTO    outd1               "DATA OUT?
        RTN                             "IGNORE OTHERS
"ENTERED FROM CODEPARS THROUGH SENDOUTP
OUTQASL LDU     PARLINE
        IFUEQ   0
        RTN                     "NO OUTPUT NOW
        LDUDA   [4]U                    "GET FLAG
        IFUEQ   0
        GOTO    chkq            "LINE IDLE?
send    SCHED   OUTQASL
        RTN
chkq    INCD    0, POCNT, setp          "HAVE PARS?
        RTN                             "NO
setp    CALL    OUTASL                   "BUILD CHARS FOR PARCEL
        LDV     PARLINE                 "V=A(LINE DESC)
        LDXY    XY[16], A[OUTBUF], 9          "A(DATA),COUNT
outp    LDXY    XY[17], 1, A[asldne]        "FLAG, COMPLETION EXIT
        SMVXYD  [2]V, XY[16], 2           "FILL IN LINE DESC
        LDUDA   [1]V                    "GET LINE CODE
        LDV     [0]U                     "PUT IT IN V
        LDU     '47'O                        "ENABLE TXRDY
        SNDCMD  [0]V                    "SEND IT
        RTN                             "DONE
"INTERRUPT RESPONSE ROUTINE
asldne  LDU     0
        STUX    lflag
        INCD    0, POCNT, send          "MORE PARS IN QUEUE
        RTN                             "NO
"CONVERT ONE PACKED PARCEL TO BASE 64 CODES
OUTASL  MOVEDY  POPTR, BA, 3      "GET NEXT PARCEL
        INCD    -1, POCNT, NEXT1         "DEC PAR CNT
NEXT1   LDU     POPTR                    "GET PTR
        ADDU    PRLEN                    "UPDATE IT
        IFUEQ   POLIM
        LDU     POBSE                    "WRAP IF NEEDED
        STUD    POPTR                    "STORE NEW PTR
        LDU     playn                    "START OF PARCEL CODE
        LDV     OUTBUF+9
out3    STVX    FA                       "SET POINTER
        STUDA   [-9]V                    "SET CODE CHAR
        LDVY    BA+2                     "LAST 16 BITS
        CALL    next                     "OUTPUT C7
        CALL    next                     "OUTPUT C6
        CALL    next                     "C5 LEFT
        STUX    FA+1                     "SAVE IT
        LDVY    BA+1                     "NEXT 16 BITS
        MULTV   4                        "GET 2 BITS IN U
```

```
          IFULT   0
          ADDU    4               "CLEAR ANY SIGN EXT.
          ADDUX   FA+1                    "COMBINE WITH FIRST 4 BITS
          STUDA   FA                    "OUTPUT C5
          CALL    next                    "OUTPUT C4
          CALL    next                    "OUTPUT C3
          CALL    next                    "C2 LEFT
          STUX    FA+1                    "TEMP SAVE
          LDVY    BA                      "FIRST 16 BITS OF PARCEL
          MULTV   16                      "GET 4 BITS IN U
          IFULT   0
          ADDU    16                      "CLEAR ANY SIGN EXT.
          ADDUX   FA+1                    "COMBINE WITH FIRST 2 BITS
          STUDA   FA                      "OUTPUT C2
          CALL    next                    "OUTPUT C1
next      ADDAX   -1, FA                  "DEC PTR
          MULTV   64                      "6 BITS
          IFULT   0
          ADDU    64                      "CLEAN OFF SIGN EXT.
          ADDU    '40'0                   "ADD ZONE
          STUDA   FA                      "OUTPUT
          RTN

saddr   "SET ADDRESS FOR DATA TRANSFER
          LDU     BOS                     "A(BUFFER)
          LDXY    rexits, A[ends], A[parinc]      "END WORD, CHAR EXITS
          GOTO    indat

data1     LDU     BOS                     "DESTINATION ADDRESS
          LDXY    rexits, A[rtn], A[parinc]       "WORD, CHAR EXITS
          GOTO    indat

inpars  "PARAMETER INPUT COLLECTION
          LDU     PRIIN                   "A(BUFFER)
          LDXY    rexits, A[end], A[parinc]       "WORD, CHAR EXITS
indat     STUDA   [rdesc]V                "SET A(BUFFER)
          LDU     1
          STUDA   [rdesc+1]V               "INITIAL SHIFTER
          LDU     0
          STUDA   [rdesc+3]V              "CLEAR ACCUM.
          SMVXYD  [6]V, rexits, 1         "STORE NEW RCV EXITS
rtn       RTN

parinc    IFUGT   '137'0
          GOTO    chke                    "HAVE CODE?
          MOVEDXY [rdesc]V, ldesc, 2      "GET RCV DESC.
          SUBU    '40'0                   "REMOVE ZONE FROM CHAR
          MULTU   shifter                 "SHIFT INTO PLACE
          ADDVY   Y[accum]                "ADD ON OLD
          STUVXY  accum                   "SAVE RESULT
          LDUY    shifter                 "GET SHIFTER
          MULTU   64                      "MAKE IT 6 MORE PLACES
          STVY    shifter                 "STORE IT
          LDV     ldesca                  "RECOVER A(LINE DESC)
          IFUEQ   0
          GOTO    pcex                    "RIGHT WORD NOT FULL?
          STUY    shifter                 "NEW SHIFTER
          LDUY    Y[accum]                "GET WORD
          STUDA   outptr                  "OUTPUT IT
          ADDAX   1, outptr               "UPDATE PTR
```

```
         LDUX    X[accum]                "GET RESIDUAL BITS
         STUY    Y[accum]                "SAVE IN RIGHT HALF
         GOTO    X[rexits]               "END OF WORD PROCESS

chke     CALL    ends                    "RESET INPUT EXIT
         GOTO    RCVINT                  "GOT PROCESS CODE

end      LDUY    shifter                 "GET SHIFTER
         IFUGT   1
         GOTO    pcex                    "NOT DONE YET?
         INCD    1, PRICNT, end2         "UPDATE QUEUE COUNT
end2     LDU     PRIIN                   "GET INPUT PTR
         IFUEQ   PRIPTR
         SCHED   SYNINP                  "WAS QUEUE EMPTY?
         ADDU    PRLEN                   "UPDATE PTR
         IFUEQ   PRILIM
         LDU     PRIBSE                  "WRAP IF NECESSARY
         STUD    PRIIN                   "STORE PTR
         IFALT   PRICNT, prmax-1, ends   "QUEUE NOT FULL?
         SETD    '23'O, xchr             "MUST SEND XOFF
         SCHED   sxo
ends     LDU     RCVINT                  "TOP LEVEL CHAR INPUT
         STUDA   [7]V                    "SET IN LINE DESC
pcex     SMVXYD  [rdesc]V, ldesc, 2      "SAVE RCV DESC
         RTN                             "EXIT

"INPUT WHEN PARCEL REMOVED FROM QUEUE
RLSEIAL  LDU     PRICNT
         IFUEQ   prmax-2
         GOTO    sxon                    "WAS QUEUE FULL?
         RTN

sxon     SETD    '21'O, xchr             "WILL SEND XON
sxo      LDU     PARLINE
         LDUDA   [4]U                    "GET FLAG
         IFUEQ   O
         GOTO    chkx                    "LINE IDLE?
         SCHED   sxo                     "NO, TRY LATER
         RTN

chkx     LDU     xchr
         LDV     1                       "ONE CHAR TO SEND
         STUVXY  XY[16]
         LDV     PARLINE                 "A(LINE DESC)
         GOTO    outp

freqn1   SETD    POIN, POPTR             "SET QUEUE EMPTY
         SETD    O, POCNT
         STVD    PARLINE                 "SET OUTPUT LINE
         RTN

stop1    SETD    O, PARLINE              "CLEAR OUTPUT LINE
         RTN

outd1    STVD    datline                 "SAVE A(LINE DESC)
coutd    LDV     datline                 "GET A(LINE DESC)
         LDUDA   [4]V                    "GET FLAG
         IFUEQ   O
         GOTO    doout                   "LINE FREE?
         SCHED   coutd                   "NO, CHECK LATER
```

```
        RTN

doout   MOVEDY  BOS, BA, 3              "GET 3 WORDS
        INCD    3, BOS, doout2          "UPDATE ADDRESS
doout2  LDU     data                    "CODE FOR BLOCK
        LDV     doutb+9                 "END OF BUFFER
        CALL    out3                    "BUILD OUTPUT CHARS FOR 3 WORDS
        LDV     datline                 "GET A(LINE DESC)
        LDXY    XY[16], A[doutb], 9             "ADDRESS, COUNT
        GOTO    outp                    "SEND BLOCK

rdesc   EQU     LINEOR-LINEO
datline EQU     OUTBUF+9
doutb   EQU     datline+1
xchr    EQU     doutb+9
prmax   EQU     20
"       LINE DESCRIPTOR FORMAT
"       0:  TXRDY+RCVRDY MASK, LINE CODE
"       2:  A(NEXT OUTPUT CHAR), COUNT
"       4:  LINE FLAG, OUTPUT DONE EXIT
"       6:  RECEIVE WORD EXIT, RECEIVE CHAR EXIT
"       RECEIVE DESCRIPTOR FORMAT
"       0:  A(NEXT BUFFER SPACE), MULTIPLIER
"       2:  DOUBLEWORD INPUT ACCUMULATOR
"       4:  4 WORDS UNUSED
        END
```